UNIVERSIDAD DE CANTABRIA

TESIS DOCTORAL

MODELOS PROBABILÍSTICOS PARA UTILIZACIÓN EN SISTEMAS EXPERTOS

Presentada por: ELENA ÁLVAREZ SÁIZ
Dirigida por: ENRIQUE CASTILLO RON

Santander, Diciembre 1989

APÉNDICE

7.-APÉNDICE

7.1.-LISTADO DEL PROGRAMA DE LA CONCHA RSPS

Unidad SELECTIONLIST

```
UNIT SelectionList;
INTERFACE
   USES
      listManager;

   PROCEDURE CreateList (n : integer;
                         list : ptr;
                         listRect : rect;
                         Sel_inicial : integer);
{SelectionList=0   Se ha pulsado un control el handle al control va en elControl}
{SelectionList=n   Seleccionado elemento de list}
   FUNCTION SelectionList (VAR elControl : ControlHandle) : integer;
   PROCEDURE closeList;
   FUNCTION newWindow2 (hor, ver : integer; titulo : str255) : windowPtr;
   FUNCTION newControl2 (h1, v1, h2, v2 : integer; titulo : STRING) : controlHandle;

IMPLEMENTATION
   TYPE
      strptr = ^str255;
      ListPtrs = ARRAY[1..1000] OF strptr;
      SelectionListPtr = ^ListPtrs;
   VAR
      myList : listHandle;
      theList : SelectionListPtr;
      theRect : rect;
      Inicio : boolean;
      Seleccion_inicial : integer;

   FUNCTION newWindow2;
      VAR
         r : rect;
         ph, pv : integer;
   BEGIN
      WITH screenBits.bounds DO
         BEGIN
            pH := (right DIV 2) - (hor DIV 2);
            pV := ((bottom DIV 2) - (ver DIV 2));
            SetRect(r, ph, pv + 15, ph + hor, pv + ver + 15);
         END;
      newWindow2 := NewWindow(NIL, r, titulo, true, 0, windowPtr(-1), false, 0);
      setPort(frontWindow);
   END;

   FUNCTION newControl2;
      VAR
         r : rect;
   BEGIN
      SetRect(r, h1, v1, h2, v2);
```

```
    newControl2 := NewControl(frontWindow, r, titulo, true, 0, 0, 255, 0, 0);
END;

PROCEDURE CreateList;
    VAR
        dataBounds : rect;
        cSize : point;
        i : integer;
        theCell : cell;
        myWindow : windowPtr;
BEGIN
    SetCursor(Arrow);
    theList := SelectionListPtr(List);
    theRect := ListRect;
    inSetRect(theRect, -1, -1);
    WITH ListRect DO
        SetRect(ListRect, left, top, right - 16, bottom);
    SetRect(dataBounds, 0, 0, 1, n);
    SetPt(cSize, 0, 0);
    myList := LNew(ListRect, dataBounds, cSize, 0, frontWindow, true, false, false, true);
    myList^^.selFlags := lOnlyOne + lExtendDrag + lNoDisjoint + lNoExtend + lNoRect + lNoNilHilite;
    FOR i := 1 TO n DO
        BEGIN
            SetPt(theCell, 0, i - 1);
            LAddToCell(ptr(longint(theList^[i]) + 1), length(theList^[i]^), theCell, myList);
        END;
    inicio := true;
    seleccion_inicial := sel_inicial;
    IF seleccion_inicial <> 0 THEN
        BEGIN
            SetPt(theCell, 0, seleccion_inicial - 1);
            LSetSelect(true, theCell, myList);
            LAutoScroll(myList);
        END;
END;

FUNCTION selectionList;
    VAR
        done : boolean;
        queControl : controlHandle;
        theEvent : eventRecord;
        myWindow : windowPtr;
        theCell : cell;
        old : integer;
BEGIN
    SetPt(theCell, 0, 0);
    IF LGetSelect(true, theCell, myList) THEN
        old := theCell.v
    ELSE
        old := -1;
    done := false;
    myWindow := frontWindow;
    IF inicio AND (seleccion_inicial <> 0) THEN
        BEGIN
            done := true;
            selectionList := seleccion_inicial;
        END;
    inicio := false;
    IF NOT done THEN
        REPEAT
            IF GetNextEvent(EveryEvent, theEvent) THEN
                CASE theEvent.what OF
```

```
                updateEvt :
                    IF (windowPtr(theEvent.message) = myWindow) THEN
                        BEGIN
                            setPort(myWindow);
                            beginUpdate(myWindow);
                            drawControls(myWindow);
                            lUpdate(myWindow^.VisRgn, myList);
                            frameRect(theRect);
                            endUpdate(myWindow);
                        END;
                MouseDown :
                    BEGIN
                        globalToLocal(theEvent.where);
                        IF LClick(theEvent.where, theEvent.modifiers, myList) THEN
                            ;
                        IF FindControl(theEvent.where, frontWindow, queControl) <> 0 THEN
                            IF TrackCOntrol(queControl, theEvent.where, NIL) <> 0 THEN
                                BEGIN
                                    elControl := queControl;
                                    selectionList := 0;
                                    done := true
                                END;
                        SetPt(theCell, 0, 0);
                        IF LGetSelect(true, theCell, myList) THEN
                            IF theCell.v <> old THEN
                                BEGIN
                                    SelectionList := theCell.v + 1;
                                    elControl := NIL;
                                    done := true;
                                END;
                    END;
                OTHERWISE
            END;
        UNTIL done;
    END;

    PROCEDURE closeList;
    BEGIN
        LDispose(myList);
    END;
END.
```

Unidad Skelglobals

```
UNIT SkelGlobals;

INTERFACE

   CONST
      iSkelDoc = 300;          { resource ID of skeletal document icon }
      NmaxSintomas = 150;
      NMaxSintomasporEnfermedad = 11;
      NMaxEnfermedades = 100;
      NMaxFrecuencias = 1024;
      NListEnfermedades = 5;
      Solutiontype = 3;

      cDiagnosis = 400;
      cDiagAuto = 401;
      cSintEnfer = 402;
      Cinicializar = 403;
      cExplicar = 404;

      cInfo = 500;
      cEnfer = 501;
      cSinto = 502;
      cHelp = 503;

      cActBase = 203;
      cIncEnferm = 204;
      cInicfromfile = 205;
      CInicSintomas = 206;
      CInicEnfermedades = 207;
      CInicFrecuencias = 208;
      CInicPopulation = 209;
      Quit = 11;

      cIdioma = 600;
      cEspanol = 601;
      cIngles = 602;
      cHabla = 604;

      cCriterioDiagnostico = 700;
      cProbabilidades = 701;
      cverosimilitudes = 702;
      cSintomasNoRelevantes = 703;

      cSimulacion = 800;
      cSimularEnfermo = 801;
      cMostrarEnfermedad = 802;
      cTodosSintomas = 803;

      FileDialogId = 500;
      SintomaDialogId = 400;
      ProbabilidadDialogId = 300;
      NoteAlertID = 300;
      Escriturawindow = 600;
      TitleProbabilidadDialogId = 12;
      Factorpopulation = 10000;
      HelpdialogId = 700;
      QuestionCursor = 500;
      PopulationDialogID = 265;
```

```
    docType = 'ECR1';
    applType = 'ECR ';

  TYPE
    ProbsintomaTypePtr = ^ProbsintomaType;
    ProbsintomaTypeHdl = ^ProbsintomaTypePtr;
    ProbsintomaEnfType = ARRAY[1..NMaxEnfermedades] OF ProbsintomaTypeHdl;
    ProbSintomaType = ARRAY[1..NMaxSintomas] OF extended;
    NumSintomasEnfermedadType = integer;
    NombreSintomasType = STRING[160];
    NombreEnfermedadType = STRING[40];
    OcurrenciasEnfermedadType = longint;
    SintomasType = ARRAY[1..NMaxSintomasporEnfermedad] OF Integer;
    SintomaPtr = ^SintomasType;
    SintomaHdl = ^SintomaPtr;
    SintomasEnfermedadType = ARRAY[1..NMaxEnfermedades] OF SintomaHdl;
    ProbType = longint;
    FrecuenciaType = ARRAY[1..NMaxFrecuencias] OF ProbType;
    FrecuenciaTypePtr = ^FrecuenciaType;
    FrecuenciaTypeHdl = ^FrecuenciaTypePtr;
    FrecuenciaCasoType = ARRAY[1..NMaxEnfermedades] OF FrecuenciaTypeHdl;
    PrType = ARRAY[1..NmaxSintomas] OF extended;
    SintomasEnfermoType = ARRAY[1..NmaxSintomas] OF integer;
    Option = 0..3;
    SintomaEnfermoDefType = ARRAY[1..NmaxSintomas] OF option;
    ProType = ARRAY[1..NMaxEnfermedades] OF ProbType;
    OrdenType = ARRAY[1..NMaxEnfermedades] OF integer;
    NumSintomasEnfermedad = ARRAY[1..NMaxEnfermedades] OF
NumSintomasEnfermedadType;
    NombreSintomas = ARRAY[1..NmaxSintomas] OF NombreSintomasType;
    NombreEnfermedad = ARRAY[1..NMaxEnfermedades] OF NombreEnfermedadType;


  VAR
    TalkOn : boolean;
    ProbSintoma : ProbsintomaEnfType;

    NumberofParameters : longint;
    EndDiagnostico, RequiredDiagnostico, Diagnosisallowed : boolean;
    Inicializado : boolean;
    NumSintomasEnfermedadPt : ^NumSintomasEnfermedad;
    NombreSintomasPt : ^NombreSintomas;
    NombreEnfermedadPt : ^NombreEnfermedad;
    SintomasEnfermedad : SintomasEnfermedadType;
    NumSintomasEnfermo, NumSintomasContraste : integer;
    NumSintomasComunes, NumSintomasComunes1, NumSintomasNoComunes : integer;
    NumSintomas : integer;
    SintomaComun, SintomaNoComun : SintomasEnfermoType;
    SintomaEnfermo, SintomaContraste : SintomasEnfermoType;
    SintomaEnfermoDef, SintomaComunDef, SintomaNoComunDef, SintomaContrasteDef :
SintomaEnfermoDefType;
    SintomaEnfermoClass : ARRAY[1..NmaxSintomas] OF option;
    Ocurrencias : Longint;
    Frecuencia : FrecuenciaCasoType;
    PEnfermedad : ProType;
    Prob : ARRAY[1..NMaxEnfermedades] OF extended;
    I, J, K, S1 : Integer;
    S : extended;
    Readend : boolean;
    RRect : Rect;
    NumeroEnfermedades : integer;
```

```
        Pr : PrType;
        Sintoma : SintomasType;
        Nulo : extended;
        ff, ff1 : text;
        L, L1, L2, TotalSize : LongInt;
        BasePopulation : Longint;
        NumCases : ProbType;
        SB : boolean;
        OcurrenciasEnfermedad : ARRAY[1..NMaxEnfermedades] OF OcurrenciasEnfermedadType;
        Def : STRING[1];
        K1, K2, K3 : LongInt;
        finput, foutput : text;
        Filename1, Filename2 : STR255;
        Nu : ordenType;
        SintomasdialogP : DialogPtr;
        Sintomasdialogdisposed : boolean;
        FrecuenciaPtr : FrecuenciaTypePtr;
        SintomaEnfermedadPtr : SintomaPtr;
        OptionProbabilidad, OptionVerosimilitud, OptionHelp, OptionSimulation,
OptionProbSintomasNoRelevantes : boolean;
        Str, str1, str2, str3 : STR255;
        Enfermosimulado : boolean;
        Enfermedadsimulada : integer;
        HelpDialog : DialogPtr;
        Cancel : boolean;
        Handlesdisposed : boolean;
        idioma : (espanol, ingles);
        MenuIngles, MenuEspanol : handle;
        HelpStrID, TxtStrID : integer;
        EnfermedadOrden, OrdenEnfermedad : ARRAY[1..NMaxEnfermedades] OF integer;

        FileResul : text;

IMPLEMENTATION


END.
```

Unidad TALK

```
UNIT talk;

INTERFACE
   USES
      SpeechIntf, SkelGlobals;

   PROCEDURE say (what : str255);
   PROCEDURE initTalk;
   PROCEDURE closeTalk;
   PROCEDURE sayText (TextPointer : ptr; textLength : longint);

IMPLEMENTATION
   VAR
      speechHdl : SpeechHandle;
      phonemeHdl : Handle;

   PROCEDURE initTalk;
      VAR   err : integer;
   BEGIN
      err := SpeechOn('', speechHdl);
      phonemeHdl := NewHandle(0);
   END;

   PROCEDURE say;
      TYPE
         texto = PACKED ARRAY[1..2048] OF char;
         textoPtr = ^Texto;
      VAR
         miTexto : textoPtr;
         err : integer;
   BEGIN
      IF (Idioma <> espanol) AND Talkon THEN
         BEGIN
            miTexto := TextoPtr(NewPtr(length(what) - 1));
            BlockMove(ptr(longint(@what) + 1), ptr(miTexto), length(what) - 1);
            err := Reader(speechHdl, ptr(miTexto), longint(length(what) - 1), phonemeHdl);
            err := MacinTalk(speechHdl, phonemeHdl);
            disposPtr(ptr(miTexto));
         END;
   END;

   PROCEDURE sayText;
      VAR   err : integer;
   BEGIN
      IF (Idioma <> espanol) AND Talkon THEN
         BEGIN
            err := Reader(speechHdl, TextPointer, textLength, phonemeHdl);
            err := MacinTalk(speechHdl, phonemeHdl);
         END;
   END;

   PROCEDURE closeTalk;
   BEGIN
      SpeechOff(speechHdl);
   END;
END.
```

Unidad INICIALIZAR

```
UNIT Inicializar;

INTERFACE

   USES
      SKelGlobals;

   PROCEDURE setdialogtext (Dialog : dialogPtr; itemNo, strNum : integer);
   PROCEDURE Alerta (Str : Str255);
   PROCEDURE InicializarDatosSintomas;
   PROCEDURE InicializarDatosEnfermedades;
   FUNCTION Estasintoma (Str : NombresintomasType) : boolean;
   FUNCTION Estaenfermedad (Str : NombreenfermedadType) : boolean;
   PROCEDURE SetNum (s : DialogPtr; N, Sloc : integer);
   PROCEDURE SetNombre (s : DialogPtr; Str1 : str255; Sloc : integer);
   PROCEDURE GetNum (s : DialogPtr; VAR N : integer; Sloc : integer);
   PROCEDURE GetNombre (s : DialogPtr; VAR str1 : str255; Sloc : integer);
   FUNCTION EnfermedadSintoma (N : integer) : integer;
   FUNCTION SintomaenEnfermedad (I, N : integer) : integer;
   PROCEDURE inicializarpoblacion;
   PROCEDURE OrdenarEnfermedades;

IMPLEMENTATION

   VAR
      Nsintoma, sitem, stype, dum : integer;
      DialogOver : boolean;
      cRect : rect;
      sHdl : Handle;
      Sdialog : Dialogptr;

   PROCEDURE trim (VAR nombre : STRING);
      VAR
         I, J, L : integer;
         done : boolean;

   BEGIN
      done := false;
      L := length(nombre);
      IF L <> 0 THEN
         BEGIN
            I := L;
            WHILE (I <> 0) AND NOT done DO
               IF nombre[I] = ' ' THEN
                  I := I - 1
               ELSE
                  done := true;
            done := false;
            J := 1;
            WHILE (J <= L) AND NOT done DO
               IF nombre[J] = ' ' THEN
                  J := J + 1
               ELSE
                  done := true;
            nombre := copy(nombre, J, I - J + 1);
         END;
   END;
```

```
PROCEDURE OrdenarEnfermedades;

   VAR
      I, J, K : integer;
      Cnombre : NombreEnfermedad;
      Min, B : NombreEnfermedadType;
      C : integer;
      reloj : cursHandle;

BEGIN
   reloj := GetCursor(WatchCursor);
   SetCursor(reloj^^);
   FOR I := 1 TO NumeroEnfermedades DO
      BEGIN
         CNombre[I] := NombreEnfermedadPt^[I];
         EnfermedadOrden[I] := I;
      END;
   FOR I := 1 TO NumeroEnfermedades DO
      BEGIN
         Min := CNombre[I];
         K := I;
         FOR J := I + 1 TO NumeroEnfermedades DO
            IF CNombre[J] < Min THEN
               BEGIN
                  Min := CNombre[J];
                  K := J;
               END;
         B := CNombre[I];
         CNombre[I] := CNombre[K];
         CNombre[K] := B;
         C := EnfermedadOrden[I];
         EnfermedadOrden[I] := EnfermedadOrden[K];
         EnfermedadOrden[K] := C;
      END;
   FOR I := 1 TO NumeroEnfermedades DO
      BEGIN
         OrdenEnfermedad[EnfermedadOrden[I]] := I;
{writeln(' i,EnfermedadOrden[I]=', i, EnfermedadOrden[I]);}
      END;
   SetCursor(Arrow);
END;

PROCEDURE Inicializarpoblacion;
   VAR
      Sdialog : DialogPtr;
      Stype, Sitem : integer;
      Srect : Rect;
      shdl : Handle;
      DialogOver : boolean;
      S, S1, Max : longint;
      I, J : integer;
      reloj : cursHandle;

BEGIN
   reloj := GetCursor(WatchCursor);
   SetCursor(reloj^^);
   S := 0;
   Max := 0;
   FOR I := 1 TO numeroenfermedades DO
      BEGIN
         S1 := 0;
         FOR J := 1 TO OcurrenciasEnfermedad[I] DO
```

```
                BEGIN
                    S1 := S1 + Frecuencia[I]^^[J];
                END;
            S := S + S1;
            IF S1 > Max THEN
                Max := S1;
        END;
    S1 := trunc(Basepopulation / Factorpopulation + 0.5);
    Max := trunc(Max / Factorpopulation + 0.5);
    S := trunc(S / Factorpopulation + 0.5);
    IF (S1 < Max) THEN
        S1 := Max;
    IF S1 > S THEN
        S1 := S;
    Sdialog := GetNewDialog(PopulationDialogID, NIL, pointer(-1));
    SetDialogText(Sdialog, 3, 72);
    GetDItem(Sdialog, 3, stype, shdl, srect);
    GetIText(shdl, Str);
    Str := concat(str, ' (', stringof(Max, ',', S, ')'));
    SetIText(shdl, Str);
    SetDialogText(Sdialog, 2, 30);
    GetDItem(Sdialog, 4, stype, shdl, srect);
    SetItext(shdl, stringof(S1));
    DialogOver := false;
    SetCursor(Arrow);
    REPEAT
        ModalDialog(NIL, sitem);
        CASE sitem OF
            1 :
                BEGIN
                    GetDItem(Sdialog, 4, stype, shdl, srect);
                    GetItext(shdl, Str);
                    trim(str);
                    IF length(Str) <> 0 THEN
                        BEGIN
                            ReadString(Str, S1);
                            IF (S1 < Max) THEN
                                BEGIN
                                    alerta(stringof('Error ', S1, ' < ', MAX));
                                    SetItext(shdl, stringof(trunc(Basepopulation / Factorpopulation +
0.5)));
                                END
                            ELSE IF S1 > S THEN
                                BEGIN
                                    alerta(stringof('Error ', S1, ' > ', S));
                                    SetItext(shdl, stringof(trunc(Basepopulation / Factorpopulation +
0.5)));
                                END
                            ELSE
                                BEGIN
                                    Basepopulation := S1 * Factorpopulation;
                                    DialogOver := true;
                                END;
                        END;
                END;
            2 :
                BEGIN
                    DialogOver := true;
                END;
            OTHERWISE
        END;
    UNTIL DialogOver;
```

```
    Disposdialog(sdialog);
END;

PROCEDURE setdialogtext;
    VAR
        item : handle;
        itemType : integer;
        r : rect;
        str : str255;
BEGIN
    GetDItem(dialog, itemno, itemType, item, r);
    getIndString(str, TxtStrID, strNum);
    IF (itemType = btnCtrl + ctrlItem) OR (itemType = btnCtrl + ctrlItem) OR (itemType = chkCtrl +
ctrlItem) OR (itemType = radCtrl + ctrlItem) THEN
        setCtitle(controlHandle(item), str)
    ELSE IF (itemType = statText) OR (itemType = editText) OR (itemType = statText + itemDisable)
OR (itemType = editText + itemDisable) THEN
        SetIText(item, str);
END;

PROCEDURE Alerta;
BEGIN
    ParamText('', str, '', '');
    dum := StopAlert(NoteAlertId, NIL);
    SysBeep(5);
END;

PROCEDURE EnableItem (s : DialogPtr;
                      Item : integer;
                      estado : boolean);

BEGIN
    GetDItem(s, Item, stype, sHdl, cRect);
    IF estado THEN
        HiliteControl(controlHandle(sHdl), 0)
    ELSE
        HiliteControl(controlHandle(sHdl), 255);
END;

FUNCTION SintomaenEnfermedad;
    VAR
        J, K : integer;
        Done : boolean;
BEGIN
    K := 0;
    IF (I > 0) AND (I <= Numeroenfermedades) AND (N > 0) AND (N <= Numsintomas) THEN
        BEGIN
            Done := false;
            Hlock(handle(SintomasEnfermedad[I]));
            J := 1;
            WHILE (J <= NumSintomasEnfermedadPt^[I]) AND NOT done DO
                BEGIN
                    IF SintomasEnfermedad[I]^^[J] = N THEN
                        BEGIN
                            done := true;
                            K := J;
                        END;
                    J := J + 1;
                END;
            Hunlock(handle(SintomasEnfermedad[I]));
        END;
    SintomaenEnfermedad := K;
```

```
END;

PROCEDURE SetNum;
BEGIN
   GetDItem(s, sloc, stype, sHdl, cRect);
   str := '';
   IF N <> 0 THEN
      NumtoString(N, str);
   SetIText(sHdl, Str);
END;

PROCEDURE SetNombre;
BEGIN
   GetDItem(s, sloc, stype, sHdl, cRect);
   SetIText(sHdl, Str1);
END;

PROCEDURE GetNum;
   VAR
      N1 : longint;
BEGIN
   GetDItem(s, sloc, stype, sHdl, cRect);
   GetIText(sHdl, Str);
   stringtoNum(str, N1);
   N := N1;
END;

PROCEDURE GetNombre;
BEGIN
   GetDItem(s, sloc, stype, sHdl, cRect);
   GetIText(sHdl, Str1);
END;

FUNCTION EnfermedadSintoma;
   VAR
      I, J, K : integer;
      Done : boolean;
BEGIN
   K := 0;
   Done := false;
   I := 1;
   WHILE (I <= NumeroEnfermedades) AND NOT Done DO
      BEGIN
         Hlock(handle(SintomasEnfermedad[I]));
         FOR J := 1 TO NumSintomasEnfermedadPt^[I] DO
            IF SintomasEnfermedad[I]^^[J] = N THEN
               BEGIN
                  done := true;
                  K := J;
               END;
         Hunlock(handle(SintomasEnfermedad[I]));
         I := I + 1;
      END;
   EnfermedadSintoma := K;
END;

FUNCTION Estaenfermedad;
   VAR
      I, J, K : integer;
      Done : boolean;
      str1, str2 : str255;
BEGIN
```

```
   Estaenfermedad := False;
   Done := false;
   I := 1;
   str1 := str;
   trim(str1);
   IF length(str1) <> 0 THEN
      BEGIN
         UprString(str1, true);
         WHILE (I <= Numeroenfermedades) AND NOT Done DO
            BEGIN
               str2 := NombreenfermedadPt^[I];
               trim(str2);
               UprString(str2, true);
               IF str1 = Str2 THEN
                  BEGIN
                     done := true;
                  END;
               I := I + 1;
            END;
      END
   ELSE
      done := true;
   Estaenfermedad := done;
END;

FUNCTION EstaSintoma;
   VAR
      I, J, K : integer;
      Done : boolean;
      str1, str2 : str255;
BEGIN
   EstaSintoma := False;
   Done := false;
   I := 1;
   str1 := str;
   Trim(str1);
   IF length(str1) <> 0 THEN
      BEGIN
         UprString(str1, true);
         WHILE (I <= Numsintomas) AND NOT Done DO
            BEGIN
               str2 := NombresintomasPt^[I];
               Trim(str2);
               UprString(str2, true);
               IF Str1 = Str2 THEN
                  BEGIN
                     done := true;
                  END;
               I := I + 1;
            END;
      END
   ELSE
      done := true;
   EstaSintoma := done;
END;


PROCEDURE InicializarDatosSintomas;
   CONST
      Sintomasdialogld = 261;
      scancelar = 1;
      santerior = 2;
```

```
        ssiguiente = 3;
        sanadir = 8;
        sborrar = 7;
        ssintoma = 4;
        snumsintoma = 5;
        snombresintoma = 6;
        smodificar = 9;

    VAR
        Nsintoma, sitem, stype, dum : integer;
        DialogOver : boolean;
        cRect : rect;
        sHdl : Handle;
        StrSin, Str : str255;
        Sdialog : Dialogptr;
        actualizado : boolean;

    PROCEDURE SetNumNomS;
    BEGIN
        SetNum(sdialog, Nsintoma, snumsintoma);
        IF (Nsintoma > 0) AND (Nsintoma <= NumSintomas) THEN
            SetNombre(sdialog, NombresintomasPt^[Nsintoma], snombresintoma)
        ELSE
            SetNombre(sdialog, '', snombresintoma);
    END;

    PROCEDURE Actualizarsintomas;
    BEGIN
        SetNumNomS;
        EnableItem(sdialog, sborrar, (Nsintoma > 0) AND (Nsintoma <= Numsintomas));
        EnableItem(sdialog, santerior, (Nsintoma > 1));
        EnableItem(sdialog, ssiguiente, (Nsintoma < Numsintomas));
        strsin := '';
    END;

BEGIN
    DialogOver := False;
    Sdialog := GetNewDialog(SintomasDialogId, NIL, Pointer(-1));
    setdialogtext(sdialog, scancelar, 30);
    setdialogtext(sdialog, santerior, 61);
    setdialogtext(sdialog, ssiguiente, 62);
    setdialogtext(sdialog, sanadir, 63);
    setdialogtext(sdialog, sborrar, 64);
    setdialogtext(sdialog, ssintoma, 65);
    setdialogtext(sdialog, smodificar, 66);
    Nsintoma := 1;
    IF Numsintomas = 0 THEN
        NombresintomasPt^[Nsintoma] := '';
    strsin := '';
    enableItem(sdialog, sanadir, (length(strsin) > 0));
    EnableItem(sdialog, smodificar, (Nsintoma > 0) AND (Nsintoma <= Numsintomas) AND
(length(strsin) > 0));
    Actualizarsintomas;
    REPEAT
        actualizado := true;
        ModalDialog(NIL, sItem);
        CASE sItem OF
            scancelar :
                DialogOver := true;
            santerior :
                BEGIN
                    GetNum(sdialog, Nsintoma, snumsintoma);
```

```
        IF (Nsintoma > 1) AND (Nsintoma <= Numsintomas) THEN
            BEGIN
                Nsintoma := Nsintoma - 1;
            END;
    END;
ssiguiente :
    BEGIN
        GetNum(sdialog, Nsintoma, snumsintoma);
        IF (Nsintoma < Numsintomas) AND (Nsintoma >= 0) THEN
            BEGIN
                Nsintoma := Nsintoma + 1;
            END;
    END;
sanadir :
    BEGIN
        GetNombre(sdialog, Str, snombresintoma);
        IF NOT estasintoma(Str) THEN
            IF (Numsintomas < Nmaxsintomas) THEN
                BEGIN
                    Numsintomas := NumSintomas + 1;
                    NombresintomasPt^[NumSintomas] := Str;
                    Nsintoma := Numsintomas;
                END
            ELSE
                BEGIN
                    getIndString(Str, TxtStrID, 43);
                    Alerta(Str);
                END
        ELSE
            BEGIN
                getIndString(Str, TxtStrID, 44);
                Alerta(Str);
            END;
        EnableItem(sdialog, sanadir, false);
        EnableItem(sdialog, smodificar, false);
    END;
sborrar :
    BEGIN
        IF (Nsintoma > 0) AND (Nsintoma <= NMaxSintomas) THEN
            BEGIN
                J := EnfermedadSintoma(Nsintoma);
                IF J = 0 THEN
                    BEGIN
                        NombresintomasPt^[Nsintoma] := NombresintomasPt^[Numsintomas];
                        NumSintomas := Numsintomas - 1;
                    END
                ELSE
                    BEGIN
                        getIndString(Str, TxtStrID, 45);
                        str := concat(str, NombreEnfermedadPt^[J]);
                        Alerta(Str);
                    END;
            END;
        IF Nsintoma >= Numsintomas THEN
            Nsintoma := Numsintomas;
    END;
snumsintoma :
    BEGIN
        GetNum(sdialog, Nsintoma, snumsintoma);
        IF Nsintoma <> 0 THEN
            BEGIN
                IF (Nsintoma > 0) AND (Nsintoma <= NumSintomas) THEN
```

```
                        SetNombre(sdialog, NombresintomasPt^[Nsintoma], snombresintoma)
                    ELSE
                        BEGIN
                            getIndString(Str, TxtStrID, 46);
                            str := concat(str, stringof('(1,', Numsintomas : 4, ')'));
                            Alerta(Str);
                            SetNombre(sdialog, '', snombresintoma);
                            Nsintoma := NumSintomas;
                        END;
                END
            ELSE
                BEGIN
                    SetNombre(sdialog, '', snombresintoma);
                END;
        END;
    smodificar :
        BEGIN
            IF NOT estasintoma(Strsin) THEN
                NombresintomasPt^[Nsintoma] := Strsin
            ELSE
                BEGIN
                    getIndString(Str, TxtStrID, 44);
                    Alerta(Str);
                END;
            EnableItem(sdialog, sanadir, false);
            EnableItem(sdialog, smodificar, false);
        END;
    snombresintoma :
        BEGIN
            GetNombre(sdialog, Strsin, snombresintoma);
            EnableItem(sdialog, smodificar, (Nsintoma > 0) AND (Nsintoma <= Numsintomas)
AND (length(strsin) > 0));
            EnableItem(sdialog, sanadir, (length(strsin) > 0));
            actualizado := false;
        END;
    OTHERWISE
        actualizado := false;
    END;
    IF actualizado THEN
        actualizarsintomas;
  UNTIL DialogOver;
  disposdialog(sdialog);
END;

PROCEDURE InicializarDatosEnfermedades;
  CONST
    EnfermedadesDialogId = 260;
    scancelar = 1;
    santerior = 2;
    ssiguiente = 3;
    sanadir = 9;
    sborrar = 10;
    ssintoma = 4;
    snumsintoma = 5;
    snombresintoma = 6;
    eanterior = 7;
    esiguiente = 8;
    eanadirenfermedad = 11;
    eborrarenfermedad = 12;
    eenfermedad = 13;
    enombreenfermedad = 14;
    enumenfermedad = 15;
```

```
        eline = 16;
        erelevantes = 17;
        etodos = 18;
        emodificar = 19;
        Dialogprobsucesold = 262;
    VAR
        Nsintoma, NEnfermedad, sitem, eitem, stype, etype, dum : integer;
        DialogOver : boolean;
        cRect : rect;
        sHdl, eHdl : Handle;
        strenf, Str : str255;
        Sdialog : Dialogptr;
        NsintomaGeneral : integer;
        Todos : boolean;
        J : integer;
        actualizado : boolean;

    PROCEDURE EscribirFrecuencias (N : integer);
        VAR
            I, J, Num : integer;
            L1 : longint;
    BEGIN
        Num := OcurrenciasEnfermedad[N];
        FOR L1 := 0 TO Num - 1 DO
            BEGIN
                write(frecuencia[N]^^[L1 + 1], ' ');
                J := NumsintomasenfermedadPt^[N];
                WHILE J >= 1 DO
                    BEGIN
                        IF Bittst(@L1, 32 - J) THEN
                            write('1')
                        ELSE
                            write('0');
                        J := J - 1;
                    END;
                writeln('');
            END;
    END;

    PROCEDURE Corregir (N, Ne, opcion : integer);
        VAR
            NewOcurrencias, I, J, Totalsize, sitem, stype : integer;
            NewFrecuencia : FrecuenciaTypeHdl;
            L2, K2, Fre : longint;
            Sdi : Dialogptr;
            Prob : real;
            DialogOv, Modifyfrecuencies : boolean;
            cRect : Rect;
            sHdl : Handle;
            Stra : STR255;
            reloj : curshandle;
    BEGIN
        IF OcurrenciasEnfermedad[Ne] > 1 THEN
            BEGIN
                reloj := GetCursor(Watchcursor);
                SetCursor(reloj^^);
{EscribirFrecuencias(Ne);}
                NewOcurrencias := OcurrenciasEnfermedad[Ne];
                IF opcion = 1 THEN
                    NewOcurrencias := NewOcurrencias DIV 2
                ELSE
                    NewOcurrencias := NewOcurrencias * 2;
```

```
                NewFrecuencia := FrecuenciaTypeHdl(NewHandle(NewOcurrencias *
Sizeof(ProbType)));
                FOR J := 1 TO NewOcurrencias DO
                   NewFrecuencia^^[J] := 0;
                Hlock(Handle(Frecuencia[Ne]));
                IF opcion = 1 THEN
                   BEGIN
                      L2 := 0;
                      WHILE L2 <= OcurrenciasEnfermedad[Ne] - 1 DO
                         BEGIN
                            K2 := L2;
                            FOR I := N TO NumSintomasEnfermedadPt^[Ne] DO
                               BEGIN
                                  IF I > 1 THEN
                                     IF BitTst(@K2, 32 - I) THEN
                                        BitSet(@K2, 33 - I)
                                     ELSE
                                        BitClr(@K2, 33 - I);
                               END;
                            BitClr(@K2, 32 - NumSintomasEnfermedadPt^[Ne]);
                            L2 := L2 + 1;
                            NeWFrecuencia^^[K2 + 1] := NeWFrecuencia^^[K2 + 1] +
Frecuencia[Ne]^^[L2];
                         END;
                   END
                ELSE
                   BEGIN
                      Sdi := GetNewDialog(Dialogprobsucesold, NIL, Pointer(-1));
                      setdialogtext(sdi, 3, 67);
                      DialogOv := false;
                      Modifyfrecuencies := false;
                      Prob := 0.0;
                      REPEAT
                         ModalDialog(NIL, sitem);
                         CASE sItem OF
                            1 :
                               BEGIN
                                  dialogOv := true;
                                  Modifyfrecuencies := true;
                               END;

                            4 :
                               BEGIN
                                  GetDItem(sdi, SItem, stype, shdl, CRect);
                                  GetItext(shdl, Stra);
                                  readString(Stra, Prob);
                               END;
                            2 :
                               dialogOv := true;

                            OTHERWISE
                         END;
                      UNTIL dialogOv;
                      disposdialog(Sdi);
                      FOR K2 := 0 TO OcurrenciasEnfermedad[Ne] - 1 DO
                         BEGIN
                            L := K2;
                            Fre := Frecuencia[Ne]^^[L + 1];
                            NewFrecuencia^^[L + 1] := Trunc(Fre * (1.0 - Prob));
                            BitSet(@L, 32 - N);
                            NewFrecuencia^^[L + 1] := Trunc(Fre * Prob);
                         END;
```

```
                    END;
                HunlockHandle(Handle(Frecuencia[Ne]));
                DisposHandle(Handle(Frecuencia[Ne]));
{TotalSize := NewOcurrencias * Sizeof(ProbType);}
{Frecuencia[Ne] := FrecuenciaTypeHdl(NewHandle(TotalSize));}
                Frecuencia[Ne] := NewFrecuencia;
                OcurrenciasEnfermedad[Ne] := NewOcurrencias;
                SetCursor(Arrow);
            END;
        END;

        PROCEDURE CrearFrecuenciasNuevasEnfermedades;
            VAR
                I, J, TotalSize : integer;
                reloj : curshandle;
        BEGIN
            reloj := GetCursor(Watchcursor);
            SetCursor(reloj^^);
            FOR I := 1 TO NumeroEnfermedades DO
                BEGIN
                    IF OcurrenciasEnfermedad[I] = 1 THEN
                        BEGIN
                            TotalSize := Trunc(exp((NumsintomasEnfermedadPt^[I]) * Ln(2.0)) + 0.1);
                            Frecuencia[I] := FrecuenciaTypeHdl(NewHandle(TotalSize *
Sizeof(ProbType)));
                            OcurrenciasEnfermedad[I] := TotalSize;
                            IF OcurrenciasEnfermedad[I] = 0 THEN
                                OcurrenciasEnfermedad[I] := 1;
                            FOR J := 1 TO OcurrenciasEnfermedad[I] DO
                                Frecuencia[I]^^[J] := 0;
{EscribirFrecuencias(I);}
                        END;
                END;
            SetCursor(Arrow);
        END;

        PROCEDURE SetNumNomE;
        BEGIN
            SetNum(sdialog, NEnfermedad, enumenfermedad);
            IF (Nenfermedad > 0) AND (NEnfermedad <= NumeroEnfermedades) THEN
                SetNombre(sdialog, NombreEnfermedadPt^[Nenfermedad], enombreenfermedad)
            ELSE
                SetNombre(sdialog, '', enombreenfermedad);
        END;

        PROCEDURE SetNumNomG;
        BEGIN
            SetNum(sdialog, Nsintomageneral, snumsintoma);
            IF (Nsintomageneral > 0) AND (NsintomaGeneral <= NumSintomas) THEN
                SetNombre(sdialog, NombresintomasPt^[Nsintomageneral], snombresintoma)
            ELSE
                SetNombre(sdialog, '', snombresintoma);
        END;

        PROCEDURE SetNumNom;
        BEGIN
            SetNum(sdialog, Nsintoma, snumsintoma);
            IF Nenfermedad > 0 THEN
                BEGIN
                    IF (Nsintoma > 0) AND (Nsintoma <= NumSintomasEnfermedadPt^[Nenfermedad])
AND (NEnfermedad > 0) AND (NEnfermedad <= NumeroEnfermedades) THEN
```

```
                SetNombre(sdialog,
NombresintomasPt^[SintomasEnfermedad[NEnfermedad]^^[Nsintoma]], snombresintoma)
            ELSE
                SetNombre(sdialog, '', snombresintoma);
        END
    ELSE
        SetNombre(sdialog, '', snombresintoma);
END;

PROCEDURE CheckEnables;
    VAR
        santeriorvalid, ssiguientevalid : boolean;
        sanadirvalid, sborrarvalid : boolean;
        eanteriorvalid, esiguientevalid : boolean;
        eborrarenfermedadvalid : boolean;

BEGIN
    santeriorvalid := true;
    ssiguientevalid := true;
    sanadirvalid := true;
    sborrarvalid := true;
    eanteriorvalid := true;
    esiguientevalid := true;
    eborrarenfermedadvalid := true;
    IF NumeroEnfermedades <= 0 THEN
        BEGIN
            IF NOT todos THEN
                BEGIN
                    santeriorvalid := false;
                    ssiguientevalid := false;
                END;
            sanadirvalid := false;
            sborrarvalid := false;
            eanteriorvalid := false;
            esiguientevalid := false;
            eborrarenfermedadvalid := false;
        END;
    IF (NEnfermedad < 1) OR (NEnfermedad > Numeroenfermedades) THEN
        BEGIN
            IF NOT todos THEN
                BEGIN
                    santeriorvalid := false;
                    ssiguientevalid := false;
                END;
            sanadirvalid := false;
            sborrarvalid := false;
            eanteriorvalid := false;
            eborrarenfermedadvalid := false;
        END
    ELSE
        BEGIN
            IF NEnfermedad = 1 THEN
                eanteriorvalid := false;

            IF (NEnfermedad = Numeroenfermedades) THEN
                esiguientevalid := false;
            IF NOT todos THEN
                BEGIN
                    IF ((Nsintoma < 1) OR (Nsintoma >
NumsintomasEnfermedadPt^[Nenfermedad])) THEN
                        BEGIN
                            santeriorvalid := false;
```

```
                        sanadirvalid := false;
                        sborrarvalid := false;
                    END;

                IF NOT todos AND (Nsintoma = 1) THEN
                    santeriorvalid := false;
                IF NOT todos AND (Nsintoma = NumsintomasEnfermedadPt^[Nenfermedad])
THEN
                    ssiguientevalid := false;
                sanadirvalid := false;
            END
        ELSE
            BEGIN
                sborrarvalid := false;
            END;
    END;
    IF Todos AND ((Nsintomageneral < 1) OR (Nsintomageneral > Numsintomas)) THEN
        BEGIN
            santeriorvalid := false;
            sanadirvalid := false;
            sborrarvalid := false;
            ssiguientevalid := false;
        END;
    IF todos AND (Nsintomageneral = 1) THEN
        santeriorvalid := false;
    IF todos AND (Nsintomageneral = Numsintomas) THEN
        ssiguientevalid := false;
    enableItem(sdialog, santerior, santeriorvalid);
    enableItem(sdialog, ssiguiente, ssiguientevalid);
    enableItem(sdialog, sanadir, sanadirvalid);
    enableItem(sdialog, sborrar, sborrarvalid);
    enableItem(sdialog, eanterior, eanteriorvalid);
    enableItem(sdialog, esiguiente, esiguientevalid);
    enableItem(sdialog, eborrarenfermedad, eborrarenfermedadvalid);
END;

PROCEDURE Actualizar;
BEGIN
    IF todos THEN
        BEGIN
            SetNumNomG;
            SetNumNomE;
        END
    ELSE
        BEGIN
            SetNumNom;
            SetNumNomE;
        END;
    actualizado := true;
END;

BEGIN
    IF Numsintomas <> 0 THEN
        BEGIN
            Strenf := '';
            DialogOver := False;
            Sdialog := GetNewDialog(EnfermedadesDialogId, NIL, Pointer(-1));
            setdialogtext(sdialog, scancelar, 30);
            setdialogtext(sdialog, santerior, 61);
            setdialogtext(sdialog, ssiguiente, 62);
            setdialogtext(sdialog, sanadir, 63);
            setdialogtext(sdialog, sborrar, 64);
```

```
setdialogtext(sdialog, ssintoma, 65);
setdialogtext(sdialog, eanterior, 61);
setdialogtext(sdialog, esiguiente, 62);
setdialogtext(sdialog, eanadirenfermedad, 63);
setdialogtext(sdialog, eborrarenfermedad, 64);
setdialogtext(sdialog, eenfermedad, 68);
setdialogtext(sdialog, erelevantes, 69);
setdialogtext(sdialog, etodos, 70);
setdialogtext(sdialog, emodificar, 66);
GetDItem(sdialog, erelevantes, stype, sHdl, cRect);
SetCtlValue(ControlHandle(sHdl), 1);
GetDItem(sdialog, etodos, stype, sHdl, cRect);
SetCtlValue(ControlHandle(sHdl), 0);
Todos := False;
IF NumeroEnfermedades > 0 THEN
    BEGIN
        Nenfermedad := 1;
        Nsintoma := 1;
        NsintomaGeneral := 1;
    END
ELSE
    BEGIN
        Nenfermedad := 0;
        Nsintoma := 0;
        NsintomaGeneral := 0;
    END;
Actualizar;
Checkenables;
EnableItem(sdialog, eanadirenfermedad, false);
EnableItem(sdialog, emodificar, false);
REPEAT
    GetDItem(Sdialog, eline, stype, sHdl, cRect);
    setPort(Sdialog);
    FillRect(cRect, black);
    GetDItem(Sdialog, snombresintoma, stype, sHdl, cRect);
    insetRect(cRect, -1, -1);
    FrameRect(cRect);
    ModalDialog(NIL, sItem);
    CASE sItem OF
        scancelar :
            DialogOver := true;
        santerior :
            BEGIN
                IF todos THEN

                    Nsintomageneral := Nsintomageneral - 1
                ELSE
                    Nsintoma := Nsintoma - 1;
                actualizado := false;
            END;
        ssiguiente :
            BEGIN
                IF todos THEN
                    Nsintomageneral := Nsintomageneral + 1
                ELSE
                    Nsintoma := Nsintoma + 1;
                actualizado := false;
            END;
        snumsintoma :
            BEGIN
                IF todos THEN
                    BEGIN
```

```
                        GetNum(sdialog, Nsintomageneral, snumsintoma);
                        IF Nsintomageneral > 0 THEN
                            BEGIN
                                IF (Nsintomageneral > NumSintomas) THEN
                                    BEGIN
                                        getIndString(Str, TxtStrID, 46);
                                        str := concat(str, stringof('(1,', Numsintomas : 4, ')'));
                                        Alerta(Str);
                                        Nsintomageneral := NumSintomas;
                                    END;
                            END;
                    END
                ELSE
                    BEGIN
                        GetNum(sdialog, Nsintoma, snumsintoma);
                        IF Nsintoma > 0 THEN
                            BEGIN
                                IF Nenfermedad > 0 THEN
                                    BEGIN
                                        IF (Nsintoma >
NumsintomasEnfermedadPt^[NEnfermedad]) THEN
                                            BEGIN
                                                getIndString(Str, TxtStrID, 46);
                                                str := concat(str, stringof('(1,',
NumsintomasEnfermedadPt^[NEnfermedad] : 4, ')'));
                                                Alerta(Str);
                                                Nsintoma :=
NumsintomasEnfermedadPt^[NEnfermedad];
                                            END;
                                    END;
                            END
                    END;
                Actualizado := false;
            END;

        eanterior :
            BEGIN
                NEnfermedad := NEnfermedad - 1;
                Nsintoma := 1;
                IF Nsintoma > NumsintomasEnfermedadPt^[Nenfermedad] THEN
                    Nsintoma := NumsintomasEnfermedadPt^[Nenfermedad];
                Actualizado := false;
            END;

        esiguiente :
            BEGIN
                NEnfermedad := NEnfermedad + 1;
                Nsintoma := 1;
                IF Nsintoma > NumsintomasEnfermedadPt^[Nenfermedad] THEN
                    Nsintoma := NumsintomasEnfermedadPt^[Nenfermedad];
                Actualizado := false;
            END;

        sanadir :
            BEGIN
                K := NumSintomasenfermedadPt^[Nenfermedad];
                IF K < NMaxSintomasporEnfermedad THEN
                    BEGIN
                        J := SintomaenEnfermedad(NEnfermedad, Nsintomageneral);
                        IF (J = 0) THEN
                            BEGIN
```

```
                                    Corregir(NumSintomasenfermedadPt^[Nenfermedad] + 1,
Nenfermedad, 2);
                                    NumSintomasEnfermedadPt^[Nenfermedad] :=
NumSintomasEnfermedadPt^[Nenfermedad] + 1;
                                    SintomasEnfermedad[Nenfermedad]^^[K + 1] :=
Nsintomageneral;
{EscribirFrecuencias(NEnfermedad);}
                                END
                             ELSE
                                BEGIN
                                    getIndString(Str, TxtStrID, 47);
                                    Alerta(Str);
                                END;
                          END
                       ELSE
                          BEGIN
                             getIndString(Str, TxtStrID, 48);
                             Alerta(Str);
                          END;
                    END;

                 sborrar :
                    BEGIN
                       Corregir(Nsintoma, Nenfermedad, 1);
                       SintomasEnfermedad[Nenfermedad]^^[Nsintoma] :=
SintomasEnfermedad[Nenfermedad]^^[NumSintomasEnfermedadPt^[Nenfermedad]];
                       NumSintomasEnfermedadPt^[Nenfermedad] :=
NumSintomasEnfermedadPt^[Nenfermedad] - 1;
                       IF Nsintoma > NumSintomasEnfermedadPt^[Nenfermedad] THEN
                          Nsintoma := NumSintomasEnfermedadPt^[Nenfermedad];
                       actualizado := false;
{EscribirFrecuencias(NEnfermedad);}
                    END;
                 eanadirenfermedad :
                    BEGIN
                       IF NOT estaenfermedad(Strenf) THEN
                          IF (Numeroenfermedades < Nmaxenfermedades) THEN
                             BEGIN
                                Numeroenfermedades := Numeroenfermedades + 1;
                                NombreenfermedadPt^[Numeroenfermedades] := Strenf;
                                NumSintomasEnfermedadPt^[Numeroenfermedades] := 0;
                                Nenfermedad := Numeroenfermedades;
                                Sintomasenfermedad[NEnfermedad] :=
SintomaHdl(NewHandle(NMaxSintomasporEnfermedad * Sizeof(integer)));
                                OcurrenciasEnfermedad[Nenfermedad] := 1;
                                Nsintoma := 0;
                                ProbSintoma[NEnfermedad] :=
ProbSintomaTypeHdl(NewHandle(sizeof(ProbsintomaType)));
                                FOR J := 1 TO numsintomas DO
                                   ProbSintoma[NEnfermedad]^^[J] := 0.0;
                             END
                          ELSE
                             BEGIN
                                getIndString(Str, TxtStrID, 50);
                                Alerta(Str);
                             END
                       ELSE
                          BEGIN
                             getIndString(Str, TxtStrID, 51);
                             Alerta(Str);
                          END;
                       Strenf := '';
```

```
                        Actualizado := false;
                        EnableItem(sdialog, eanadirenfermedad, false);
                        EnableItem(sdialog, emodificar, false);
                     END;
                  eborrarenfermedad :
                     BEGIN
                        NombreenfermedadPt^[Nenfermedad] :=
NombreenfermedadPt^[Numeroenfermedades];
                        NumSintomasEnfermedadPt^[Nenfermedad] :=
NumSintomasEnfermedadPt^[Numeroenfermedades];
                        NumSintomasEnfermedadPt^[Numeroenfermedades] := 0;
                        DisposHandle(handle(SintomasEnfermedad[NEnfermedad]));
                        Sintomasenfermedad[Nenfermedad] :=
Sintomasenfermedad[Numeroenfermedades];
                        Numeroenfermedades := Numeroenfermedades - 1;
                        DisposHandle(handle(ProbSintoma[NEnfermedad]));
                        IF OcurrenciasEnfermedad[NEnfermedad] <> 1 THEN
                           DisposHandle(Handle(Frecuencia[NEnfermedad]));
                        IF NEnfermedad > Numeroenfermedades THEN
                           Nenfermedad := Numeroenfermedades;
                        Nsintoma := 1;
                        Actualizado := false;
                     END;
                  enumenfermedad :
                     BEGIN
                        GetNum(sdialog, Nenfermedad, enumenfermedad);
                        IF Nenfermedad > 0 THEN
                           BEGIN
                              IF (Nenfermedad > Numeroenfermedades) THEN
                                 BEGIN
                                    getIndString(Str, TxtStrID, 52);
                                    str := concat(str, stringof('(1,', Numeroenfermedades : 4, ')'));
                                    Alerta(Str);
                                    Nenfermedad := Numeroenfermedades;
                                 END;
                              IF Nsintoma > NumsintomasEnfermedadPt^[Nenfermedad] THEN
                                 Nsintoma := NumsintomasEnfermedadPt^[Nenfermedad];
                           END;
                        actualizado := false;
                     END;
                  erelevantes :
                     BEGIN
                        GetDItem(sdialog, erelevantes, stype, sHdl, cRect);
                        SetCtlValue(ControlHandle(sHdl), 1);
                        GetDItem(sdialog, etodos, stype, sHdl, cRect);
                        SetCtlValue(ControlHandle(sHdl), 0);
                        Todos := False;
                        GetNum(sdialog, Nsintoma, snumsintoma);
                        IF Nenfermedad > 0 THEN
                           IF Nsintoma > NumsintomasEnfermedadPt^[NEnfermedad] THEN
                              Nsintoma := 1;
                        actualizado := false;
                     END;
                  etodos :
                     BEGIN
                        GetDItem(sdialog, etodos, stype, sHdl, cRect);
                        SetCtlValue(ControlHandle(sHdl), 1);
                        GetDItem(sdialog, erelevantes, stype, sHdl, cRect);
                        SetCtlValue(ControlHandle(sHdl), 0);
                        Todos := true;
                        GetNum(sdialog, Nsintomageneral, snumsintoma);
                        IF (NsintomaGeneral = 0) AND (Numsintomas > 0) THEN
```

```
                    Nsintomageneral := 1;
                  actualizado := false;
                END;
              emodificar :
                BEGIN
                  IF NOT estaenfermedad(Strenf) THEN
                    BEGIN
                      NombreenfermedadPt^[Nenfermedad] := Strenf;
                    END
                  ELSE
                    BEGIN
                      getIndString(Str, TxtStrID, 51);
                      Alerta(Str);
                    END;
                  Strenf := '';
                  actualizado := false;
                  EnableItem(sdialog, eanadirenfermedad, false);
                  EnableItem(sdialog, emodificar, false);
                END;
              enombreenfermedad :
                BEGIN
                  GetNombre(sdialog, Strenf, enombreenfermedad);
                  EnableItem(sdialog, eanadirenfermedad, length(strenf) > 0);
                  EnableItem(sdialog, emodificar, length(strenf) > 0);
                END;
              OTHERWISE
            END;
            IF NOT actualizado THEN
              BEGIN
                Actualizar;
                Checkenables;
              END;
          UNTIL DialogOver;
          disposdialog(sdialog);
          CrearFrecuenciasNuevasEnfermedades;
        END
      ELSE
        BEGIN
          getIndString(Str, TxtStrID, 53);
          Alerta(Str);
        END;
      IF NumeroEnfermedades > 0 THEN
        inicializado := true;
  END;

END.
```

Unidad DISPLAY_LIST

```
UNIT display_list;

INTERFACE
   USES
      SkelGlobals, SelectionList, Talk;
   PROCEDURE display_list (bounds : rect; titulo : str255; elNumero : integer;
                           laInterlinea : integer; elDrawProc : procPtr);
   PROCEDURE Ordenar1 (VAR Sintomas : SintomasEnfermoType;
                           VAR SintomaEnfermoDef : SintomaEnfermoDefType;
                           N : integer);
   PROCEDURE explicar;
   PROCEDURE SintomasComunes (I, NumsintomasContraste : integer;
                           SintomaContraste : SintomasEnfermoType;
                           SintomaContrasteDef : SintomaEnfermoDeftype);
   FUNCTION Indiceordenprob (i : integer) : integer;
   PROCEDURE wait (T : integer);
   PROCEDURE limpiar (VAR str : STRING);

IMPLEMENTATION

   PROCEDURE wait;
      VAR
         I : longint;
   BEGIN
      I := Tickcount;
      REPEAT
      UNTIL (Tickcount - I) > T;
   END;

   PROCEDURE limpiar;
      VAR
         I, L : integer;
   BEGIN
      L := length(str);
      FOR I := 1 TO L DO
         IF str[I] = "" THEN
            str[I] := ' ';
   END;

   FUNCTION Indiceordenprob;
      VAR
         J : integer;
   BEGIN
      J := 1;
      WHILE Nu[J] <> I DO
         BEGIN
            J := J + 1
         END;
      Indiceordenprob := J;
   END;

   PROCEDURE SintomasComunes;
      VAR
         K, J : integer;
   BEGIN
      NumSintomasComunes := 0;
      NumSintomasNoComunes := 0;
      K := 1;
```

```
      J := 1;
      REPEAT
         IF (SintomasEnfermedad[I]^^[K] = SintomaContraste[J]) AND (SintomaContrasteDef[J] <> 3)
THEN
            BEGIN
               NumSintomasComunes := NumSintomasComunes + 1;
               SintomaComun[NumSintomasComunes] := K;
               SintomaComunDef[NumSintomasComunes] := SintomaContrasteDef[J];
               K := K + 1;
               J := J + 1;
            END
         ELSE
            BEGIN
               IF (SintomasEnfermedad[I]^^[K] < SintomaContraste[J]) AND (K <
NumSintomasEnfermedadPt^[I]) THEN
                  K := K + 1
               ELSE
                  BEGIN
                     NumSintomasNoComunes := NumSintomasNoComunes + 1;
                     SintomaNoComun[NumSintomasNoComunes] := SintomaContraste[J];
                     SintomaNoComunDef[NumSintomasNoComunes] := SintomaContrasteDef[J];
                     J := J + 1;
                  END;
            END;
      UNTIL (K > NumSintomasEnfermedadPt^[I]) OR (J > NumSintomasContraste);
      IF J <= NumsintomasContraste THEN
         FOR K := J TO NumsintomasContraste DO
            BEGIN
               NumSintomasNoComunes := NumSintomasNoComunes + 1;
               SintomaNoComun[NumSintomasNoComunes] := SintomaContraste[K];
               SintomaNoComunDef[NumSintomasNoComunes] := SintomaContrasteDef[K];
            END;
      IF false THEN
         BEGIN
            write('Enfermedad=', I, ' Sintomas comunes : ');
            FOR J := 1 TO NumSintomasComunes DO
               write(SintomaComun[J], '  def=', SintomaComunDef[J]);
            writeln(' ');
            write('Enfermedad=', I, ' Sintomas no comunes : ');
            FOR J := 1 TO NumSintomasNoComunes DO
               write(SintomaNoComun[J], '  def=', SintomaNoComunDef[J]);
            writeln(' ');
         END;
   END;

   CONST
      AnchoScroll = 16;
      largoBoton = 70;
   VAR
      myWindow, windPtr : windowPtr;
      windowRec : windowRecord;
      hScroll, OKButton, queControl : controlHandle;
      dataRect, r : rect;
      centro, parte, tecla, temp, temp2, numero, interlinea : integer;
      theEvent : eventRecord;
      done : boolean;
      myRgn : rgnHandle;
      drawProc : procPtr;
      savePort : GrafPtr;

   PROCEDURE call_proc (proc : procPtr);
   INLINE
```

```
      $205F,  { MOVE.L (A7)+,D0   }
      $4E90;  { JSR (D0)          }

   PROCEDURE Ordenar1;
      VAR
         B : integer;
         I, K : integer;
         C : option;
   BEGIN
      J := N - 1;
      REPEAT
         K := 0;
         FOR I := 1 TO J DO
            IF Sintomas[I] > Sintomas[I + 1] THEN
               BEGIN
                  B := Sintomas[I];
                  Sintomas[I] := Sintomas[I + 1];
                  Sintomas[I + 1] := B;
                  C := SintomaEnfermoDef[I];
                  SintomaEnfermoDef[I] := SintomaEnfermoDef[I + 1];
                  SintomaEnfermoDef[I + 1] := C;
                  K := 1;
                  J := I;
               END;
      UNTIL K = 0;
{  for I := 1 to N do}
{  write(Sintomas[I] : 4, SintomaEnfermoDef[I] : 6);}
{  writeln(' ');}
   END;

   PROCEDURE draw_content (clip : rect);
   BEGIN
      eraseRect(clip);
      WITH dataRect DO
         BEGIN
            setOrigin(-left, -top + getCtlValue(hScroll));
            offSetRect(clip, -left, -top + getCtlValue(hScroll));
            clipRect(clip);
{  setRect(r, 0, getCtlValue(hScroll), right - left, bottom - top + getCtlValue(hScroll));}
{  clipRect(r)}
         END;
      moveTo(0, 0);
      TextSize(9);
      call_proc(drawProc);
      TextSize(12);
      setOrigin(0, 0);
      clipRect(myWindow^.portRect);
   END;

   PROCEDURE handle_button;
   BEGIN
      IF trackControl(OKButton, theEvent.where, NIL) <> 0 THEN
         done := true;
   END;

   PROCEDURE myAction (queControl : controlHandle;
                       parte : integer);
   BEGIN
      temp := getCtlValue(hScroll);
      CASE parte OF
         inUpButton :
            SetCtlValue(hScroll, GetCtlValue(hScroll) - interLinea);
```

```
            inDownButton :
                SetCtlValue(hScroll, GetCtlValue(hScroll) + interLinea);
            inPageUp :
                SetCtlValue(hScroll, GetCtlValue(hScroll) - dataRect.bottom - dataRect.top);
            inPageDown :
                SetCtlValue(hScroll, GetCtlValue(hScroll) + dataRect.bottom - dataRect.top);
            OTHERWISE
                ;
        END;
        clipRect(dataRect);
        scrollRect(dataRect, 0, temp - getCtlValue(hScroll), myRgn);
        draw_content(myRgn^^.rgnBBox);
    END;

    PROCEDURE handle_scroll;
    BEGIN
        CASE parte OF
            inThumb :
                BEGIN
                    parte := trackControl(hScroll, theEvent.where, NIL);
                    draw_content(dataRect);
                END;
            OTHERWISE
                parte := trackControl(hScroll, theEvent.where, @myAction);
        END;
    END;

    PROCEDURE display_list;
    BEGIN
        flushEvents(mDownMask + keyDownMask, 0);
        getPort(savePort);
        numero := elNumero;
        interlinea := laInterlinea;
        drawProc := elDrawProc;
        r := bounds;
{setRect(r, 300, 50, 500, 250);}
        myWindow := newWindow(@windowRec, r, '', false, 1, windowptr(-1), false, 0);
        showWindow(myWindow);
        setPort(myWindow);
        WITH myWindow^.portRect DO
            setRect(r, 10, 5, right - 10, 55);
        textFace([bold]);
        TextSize(10);
        TextBox(pointer(ord(@titulo) + 1), length(titulo), r, teJustCenter);
        textFace([]);
        TextSize(12);
        WITH myWindow^.portRect DO
            setRect(r, 10, 61, right - 10, bottom - 40);
        frameRect(r);
        dataRect := r;
        WITH r DO
            setRect(r, right - AnchoScroll, top, right, bottom);
        temp := numero * interlinea - (dataRect.bottom - dataRect.top) + 5;
        IF temp < 2 THEN
            BEGIN
                temp := 0;
                temp2 := 0;
            END
        ELSE
            temp2 := 1;
        hScroll := newControl(myWindow, r, '', true, 1, temp2, temp, scrollBarProc, 0);
        WITH dataRect DO
```

```
      SetRect(dataRect, left, top, right - AnchoScroll, bottom);
   inSetRect(dataRect, 1, 1);
   WITH myWindow^.portRect DO
      BEGIN
         centro := ((right - left) DIV 2) - (largoBoton DIV 2);
         setRect(r, centro, bottom - 27, centro + largoBoton, bottom - 7);
      END;
   OKButton := newControl(myWindow, r, 'OK', true, 0, 0, 0, pushButProc, 0);
   inSetRect(r, -4, -4);
   penSize(3, 3);
   frameRoundRect(r, 16, 16);
   penSize(1, 1);
   myRgn := newRgn;
   draw_content(dataRect);
   done := false;
   setCursor(arrow);
   REPEAT
      IF getNextEvent(mDownMask + keyDownMask, theEvent) THEN
         CASE theEvent.what OF
            mouseDown :
               IF findWindow(theEvent.where, windPtr) = inContent THEN
                  IF windPtr = myWindow THEN
                     BEGIN
                        globalToLocal(theEvent.where);
                        parte := findControl(theEvent.where, windPtr, queControl);
                        IF queControl = hScroll THEN
                           handle_Scroll
                        ELSE IF queControl = OKButton THEN
                           handle_button;
                     END;
            keyDown :
               BEGIN
                  tecla := bitAnd(theEvent.message, charCodeMask);
                  IF (tecla = 13) OR (tecla = 3) THEN
                     done := true;
               END;
            OTHERWISE
               ;
         END;
   UNTIL done;
   disposeControl(okButton);
   disposeControl(hScroll);
   closeWindow(myWindow);
   CloseRgn(myRgn);
   setPort(savePort);
END;

PROCEDURE explicar;
   VAR
      texto : PACKED ARRAY[1..2048] OF char;
      myList : ARRAY[1..NMaxEnfermedades] OF ptr;
      i, j, resultado, longTexto : integer;
      myWindow : WindowPtr;
      r : rect;
      OK, queControl : ControlHandle;
      str : str255;
      ver : extended;
      Title : boolean;

   PROCEDURE addStrLn (str : STRING);
      VAR
         i : integer;
```

```
    BEGIN
      FOR i := 1 TO length(str) DO
        texto[longTexto + i] := str[i];
      longTexto := longTexto + length(str) + 1;
      texto[longTexto] := chr(13);
    END;

    PROCEDURE addStr (str : STRING);
      VAR
        i : integer;
    BEGIN
      FOR i := 1 TO length(str) DO
        texto[longTexto + i] := str[i];
      longTexto := longTexto + length(str);
    END;

  BEGIN
    FOR i := 1 TO NumeroEnfermedades DO
      myList[i] := @nombreEnfermedadPt^[EnfermedadOrden[I]];
    GetIndString(str, TxtStrID, 36);
    myWindow := NewWindow2(500, 280, str);
    GetIndString(str, TxtStrID, 35);
    OK := NewControl2(450, 260, 490, 275, str);
    SetRect(r, 10, 12, 180, 265);
    TextSize(9);
    CreateList(NumeroEnfermedades, @myList[1], r, OrdenEnfermedad[Nu[1]]);
    REPEAT
      resultado := SelectionList(queControl);
      IF resultado <> 0 THEN
        BEGIN
          resultado := EnfermedadOrden[resultado];
          longTexto := 0;
          setRect(r, 190, 8, 490, 239);
          eraseRect(r);
          MoveTo(195, 20);
          TextFace([Bold]);
          TextSize(12);
          DrawString(NombreEnfermedadPt^[resultado]);
          TextFace([]);
          TextSize(9);
          Ordenar1(SintomaEnfermo, SintomaEnfermoDef, NumSintomasEnfermo);
          SintomasComunes(Resultado, NumsintomasEnfermo, SintomaEnfermo,
SintomaEnfermoDef);
          ver := 0.0;
          IF Prob[1] > 0.0 THEN
            ver := Prob[Indiceordenprob(Resultado)] / Prob[1];
          Title := false;
          IF Ver > 0.3 THEN
            BEGIN
              getIndString(Str, TxtStrID, 37);
              AddStrLn(Str);
              FOR J := 1 TO NumSintomasComunes DO
                BEGIN
                  IF SintomaComunDef[J] = 1 THEN
                    BEGIN
                      IF Title = false THEN
                        BEGIN
                          AddStrIn('      ');
                          getIndString(Str, TxtStrID, 38);
                          AddStrLn(Str);
                          AddStrIn('      ');
                          Title := true;
```

```
                                    END;
{AddStr('      ');}
    AddStrLn(concat(NombreSintomasPt^[SintomasEnfermedad[Resultado]^^[SintomaComun[J]]],
'.'));
                                END;
                            END;
                        Title := false;
                        FOR J := 1 TO NumSintomasNoComunes DO
                            BEGIN
                                IF SintomaNoComunDef[J] = 2 THEN
                                    BEGIN
                                        IF Title = false THEN
                                            BEGIN
                                                AddStrIn('      ');
                                                getIndString(Str, TxtStrID, 39);
                                                AddStrLn(Str);
                                                AddStrIn('      ');
                                                Title := true;
                                            END;
{AddStr('      ');}
                    AddStrIn(Concat(NombreSintomasPt^[SintomaNoComun[J]], '.'));
                                END;
                            END;
                    END
                ELSE
                    BEGIN
                        getIndString(Str, TxtStrID, 40);
                        AddStrLn(Str);
                        FOR J := 1 TO NumSintomasComunes DO
                            BEGIN
                                IF SintomaComunDef[J] = 2 THEN
                                    BEGIN
                                        IF Title = false THEN
                                            BEGIN
                                                AddStrIn('      ');
                                                getIndString(Str, TxtStrID, 41);
                                                AddStrLn(Str);
                                                AddStrIn('      ');
                                                Title := true;
                                            END;
{AddStr('      ');}
    AddStrIn(Concat(NombreSintomasPt^[SintomasEnfermedad[Resultado]^^[SintomaComun[J]]],
'.'));
                                END;
                            END;
                        Title := false;
                        FOR J := 1 TO NumSintomasNoComunes DO
                            BEGIN
                                IF SintomaNoComunDef[J] = 1 THEN
                                    BEGIN
                                        IF Title = false THEN
                                            BEGIN
                                                AddStrIn('      ');
                                                getIndString(Str, TxtStrID, 42);
                                                AddStrLn(Str);
                                                AddStrIn('      ');
                                                Title := true;
                                            END;
{AddStr('      ');}
                                    AddStrIn(Concat(NombreSintomasPt^[SintomaNoComun[J]], '.'));
```

```
                        END;
                    END;
                END;
            SetRect(R, 198, 30, 495, 259);
            TextBox(@Texto, longTexto, r, teJustLeft);
            IF (idioma <> espanol) AND TalkOn THEN
                BEGIN
                    SayText(@Texto, longTexto);
                END;
        END;
    UNTIL (queControl = OK);
    closeList;
    closeWindow(myWindow);
  END;
END.
```

Unit EXPERTO

```
Unit Experto;

INTERFACE
   USES
      SkelGlobals, display_list, talk;

   PROCEDURE set_dialog_text (Dialog : dialogPtr; itemNo, strNum : integer);
   PROCEDURE DisposeallhdI;
   PROCEDURE Read_data (vRefNum : integer; filename2 : str255);
   FUNCTION get_file_name (VAR vRefnum : integer; VAR fname : str255) : boolean;
   PROCEDURE Information (N, Kopt : integer);
   PROCEDURE Sintomas (N : integer);
   PROCEDURE Actualizar;
   PROCEDURE Diagnostico (VAR diagnosticofailed : boolean);
   PROCEDURE orden (VAR dialogpointer : DialogPtr);
   PROCEDURE InicializarSintomas;
   PROCEDURE Dialog3 (DialogID : integer;VAR Str1, Str2 : STR255; VAR ditem : integer);
   PROCEDURE Dialog (DialogID : integer; VAR Filename3 : STR255);
   PROCEDURE PreguntaSintoma (VAR lastbutton : integer);
   PROCEDURE EnfermoaBase;
   PROCEDURE SimularEnfermo;
   PROCEDURE MostrarEnfermedad;
   PROCEDURE Errordisk (error : integer);
   PROCEDURE Ordenar (VAR Sintomas : SintomasType; VAR Pr : PrType; N : integer);
   PROCEDURE bitsdesc (L : longint; N : integer);
   PROCEDURE trim (VAR nombre : STRING);

IMPLEMENTATION
   VAR
      currEnfermedad, currSintoma, numEnferSinto : integer;
      stype : integer;
      sHdl : Handle;
      cRect : rect;

   PROCEDURE EnableItem (s : DialogPtr;
                         Item : integer;
                         estado : boolean);
   BEGIN
      GetDItem(s, Item, stype, sHdl, cRect);
      IF estado THEN
         HiliteControl(controlHandle(sHdl), 0)
      ELSE
         HiliteControl(controlHandle(sHdl), 255);
   END;

   PROCEDURE trim;
      VAR
         I, J, L : integer;
         done : boolean;
   BEGIN
      done := false;
      L := length(nombre);
      IF L <> 0 THEN
         BEGIN
            I := L;
            WHILE (I <> 0) AND NOT done DO
               IF nombre[I] = ' ' THEN
                  I := I - 1
```

```
                ELSE
                    done := true;

            done := false;
            J := 1;
            WHILE (J <= L) AND NOT done DO
                IF nombre[J] = ' ' THEN
                    J := J + 1
                ELSE
                    done := true;
            nombre := copy(nombre, J, l - J + 1);
        END;
    END;


    PROCEDURE set_dialog_text; {(Dialog : dialogPtr;}
                                {itemNo, strNum : integer);}
        VAR
            item : handle;
            itemType : integer;
            r : rect;
            str : str255;
    BEGIN
        GetDItem(dialog, itemno, itemType, item, r);
        getIndString(str, TxtStrID, strNum);
        IF (itemType = btnCtrl + ctrlItem) OR (itemType = btnCtrl + ctrlItem) OR (itemType = chkCtrl +
ctrlItem) OR (itemType = radCtrl + ctrlItem) THEN
            setCtitle(controlHandle(item), str)
        ELSE IF (itemType = statText) OR (itemType = editText) OR (itemType = statText + itemDisable)
OR (itemType = editText + itemDisable) THEN
            SetIText(item, str);
    END;

    PROCEDURE Disposeallhdl;
        VAR
            I, J : integer;
    BEGIN
        IF handlesdisposed = False THEN
            BEGIN
                Handlesdisposed := True;
                FOR I := 1 TO NumeroEnfermedades DO
                    BEGIN
                        disposHandle(handle(SintomasEnfermedad[I]));
                        disposHandle(Handle(Probsintoma[I]));
                        disposHandle(Handle(Frecuencia[I]));
                    END;
            END;
    END;

    PROCEDURE Errordisk;
        VAR
            Dum : integer;
    BEGIN
        IF error <> noErr THEN
            BEGIN
                getIndString(str, TxtStrID, 9);
                Str := Stringof(Str, error);
                ParamText('', Str, '', '');
                IF talkOn AND (idioma <> espanol) THEN
                    say(str);
                Dum := StopAlert(NoteAlertID, NIL);
                SysBeep(5);
```

```
            ExitToShell;
        END;
    END;

    PROCEDURE PreguntaSintoma;
        VAR
            I, J, K, L, Lsintoma : integer;
            Sintomafound : boolean;
            Dum : integer;

        PROCEDURE Dialog1 (DialogID, Itemstop1, Itemstop2 : integer;
                                VAR lastbutton : integer);
            VAR
                DItem, CType : integer;
                CHdl : Handle;
                CRect : Rect;
        BEGIN
            IF Sintomasdialogdisposed = True THEN
                BEGIN
                    SintomasdialogP := GetNewDialog(DialogID, NIL, POINTER(-1));
                    Sintomasdialogdisposed := False;
                    lastbutton := 5;
                    GetDItem(SintomasdialogP, lastbutton, CType, CHdl, CRect);
                    SetCtlValue(ControlHandle(CHdl), 1);
                END;
            IF DialogID = SintomaDialogID THEN
                BEGIN
                    Set_dialog_text(SintomasdialogP, 5, 27);{Si}
                    Set_dialog_text(SintomasdialogP, 6, 28);{no}
                    Set_dialog_text(SintomasdialogP, 7, 29);{no sabe}
                    Set_dialog_text(SintomasdialogP, 2, 30);{cancelar}
                END;

            GetDItem(SintomasdialogP, 4, CType, CHdl, CRect);
            SetIText(CHdl, NombreSintomasPt^[Lsintoma]);
            GetDItem(SintomasdialogP, Itemstop1, CType, CHdl, CRect);
            SetPort(SintomasdialogP);
            InsetRect(CRect, -4, -4);
            PenSize(3, 3);
            FrameRoundRect(CRect, 18, 18);
            PenSize(1, 1);
            REPEAT
                ModalDialog(NIL, dItem);
                GetDItem(SintomasdialogP, lastbutton, CType, CHdl, CRect);
                IF (DItem <> Itemstop1) AND (DItem <> Itemstop2) THEN
                    BEGIN
                        SetCtlValue(ControlHandle(CHdl), 0);
                        lastbutton := dItem;
                        GetDItem(SintomasdialogP, lastbutton, CType, CHdl, CRect);
                        SetCtlValue(ControlHandle(CHdl), 1);
{writeln(ditem);}
                    END;
            UNTIL (ditem = Itemstop1) OR (ditem = Itemstop2) OR (ditem = 1);
            IF ditem = Itemstop2 THEN
                BEGIN
                    lastbutton := 0;
                    EndDiagnostico := True;
                    DisposDialog(SintomasdialogP);
                    Sintomasdialogdisposed := True;
                END;
        END;
```

```
BEGIN
   Lsintoma := 0;
   Sintomafound := False;
   I := 1;
   REPEAT
      J := Nu[I];
      K := 1;
      REPEAT
         L := SintomasEnfermedad[J]^^[K];
         IF SintomaEnfermoClass[L] = 0 THEN
            BEGIN
               Lsintoma := L;
               Sintomafound := True;
            END;
         K := K + 1;
      UNTIL (K > NumSintomasEnfermedadPt^[J]) OR Sintomafound;
      I := I + 1;
   UNTIL (I > NListEnfermedades) OR (I > Numeroenfermedades) OR Sintomafound;
   IF Lsintoma <> 0 THEN
      BEGIN
         Dialog1(SintomaDialogId, 1, 2, lastbutton);
         IF lastbutton > 4 THEN
            BEGIN
               SintomaEnfermoClass[Lsintoma] := lastbutton - 4;
               NumsintomasEnfermo := NumsintomasEnfermo + 1;
               SintomaEnfermo[NumSintomasEnfermo] := Lsintoma;
               SintomaEnfermoDef[NumSintomasEnfermo] := SintomaEnfermoClass[Lsintoma];
               IF SintomaEnfermoClass[Lsintoma] = 3 THEN
                  Requireddiagnostico := FALSE
               ELSE
                  BEGIN
                     DisposDialog(SintomasdialogP);
                     Sintomasdialogdisposed := True;
                  END;
            END
         ELSE
            EndDiagnostico := True;
      END
   ELSE
      BEGIN
         IF NOT Sintomasdialogdisposed THEN
            DisposDialog(SintomasdialogP);
         Sintomasdialogdisposed := True;
         EndDiagnostico := True;
         getIndString(str, TxtStrID, 8);
         ParamText('', Str, '', '');
         Dum := NoteAlert(NoteAlertID, NIL);
      END;
END;

PROCEDURE Dialog3;
   VAR
      CType : integer;
      DialogP : DialogPtr;
      CHdl : Handle;
      CRect : Rect;
BEGIN
   dialogP := GetNewDialog(DialogID, NIL, POINTER(-1));
   GetDItem(dialogP, 3, CType, CHdl, CRect);
   SetItext(CHdl, Str1);
   GetDItem(dialogP, 4, CType, CHdl, CRect);
   SetItext(CHdl, Str2);
```

```
    IF talkOn AND (idioma <> Espanol) THEN
        BEGIN
            DrawDialog(dialogP);
            Say(concat(str1, '.', chr(13), str2));
        END;
    REPEAT
        ModalDialog(NIL, dItem);
        GetDItem(dialogP, 4, CType, CHdl, CRect);
        GetItext(CHdl, Str2);
    UNTIL (ditem = 1) OR (ditem = 2);
    DisposDialog(dialogP);
END;

PROCEDURE Dialog;
    VAR
        DItem, CType : integer;
        DialogP : DialogPtr;
        CHdl : Handle;
        CRect : Rect;
BEGIN
    dialogP := GetNewDialog(DialogID, NIL, POINTER(-1));
    GetDItem(dialogP, 4, CType, CHdl, CRect);
    SetItext(CHdl, Filename3);
    REPEAT
        ModalDialog(NIL, dItem);
        GetDItem(dialogP, 4, CType, CHdl, CRect);
        GetItext(CHdl, Filename3);
    UNTIL (ditem = 1);
    DisposDialog(dialogP);
END;

PROCEDURE InicializarSintomas;
    VAR
        I : integer;
BEGIN
    FOR I := 1 TO NmaxSintomas DO
        SintomaEnfermoClass[I] := 0;
    Diagnosisallowed := False;
    Enfermosimulado := False;
END;

FUNCTION get_file_name; {(var vrefnum : integer;}
                        {var fname : str255) : boolean;}
    VAR
        reply : SFReply;
        pt : point;
        typeList : SFTypeList;
BEGIN
    SetPt(pt, 80, 60);
    typeList[0] := DocType;
    SFGetFile(pt, '', NIL, 1, typeList, NIL, reply);
    IF NOT reply.good THEN
        get_file_name := false
    ELSE
        BEGIN
            VRefNum := reply.vRefNum;
            fname := reply.fName;
            get_file_name := true;
        END;
END;

PROCEDURE Read_data;
```

```
    VAR
      I, J : integer;
      reloj : cursHandle;
      Temp : Str255;
      RefNum : integer;
      Error : OsErr;
      MyDialog : DialogPtr;
      TheItem : integer;
      Dum : integer;
      textr : rect;

    PROCEDURE ReadByte (what : ptr;
                        num : longInt);
    BEGIN
      error := FSREad(refNum, num, what);
      ErrorDisk(error);
    END;

  BEGIN
    reloj := GetCursor(WatchCursor);
    SetCursor(reloj^^);
    Error := FSOpen(Filename2, VRefNum, RefNum);
    ErrorDisk(error);
    ReadByte(@Basepopulation, sizeof(Basepopulation));
    ReadByte(@Numsintomas, sizeof(numSintomas));
    ReadByte(@NumeroEnfermedades, sizeof(NumeroEnfermedades));
    FOR I := 1 TO NumeroEnfermedades DO
      BEGIN
        ProbSintoma[I] := ProbSintomaTypeHdl(NewHandle(sizeof(ProbsintomaType)));
        Hlock(handle(ProbSintoma[I]));
        ReadByte(@ProbSintoma[I]^^[1], Sizeof(ProbsintomaType));
        Hunlock(handle(ProbSintoma[I]));
      END;
    ReadByte(@NombreSintomasPt^[1], sizeOf(nombreSintomasType) * NumSintomas);
    ReadByte(@NombreEnfermedadPt^[1], sizeOf(NombreEnfermedadType) *
NumeroEnfermedades);
    ReadByte(@NumSintomasEnfermedadPt^[1], sizeOf(NumSintomasEnfermedadType) *
NumeroEnfermedades);
    ReadByte(@OcurrenciasEnfermedad[1], sizeof(OcurrenciasEnfermedadType) *
NumeroEnfermedades);
    NumberofParameters := 0;
    FOR I := 1 TO NumeroEnfermedades DO
      BEGIN
        SintomasEnfermedad[I] := SintomaHdl(NewHandle(NMaxSintomasporEnfermedad *
Sizeof(Integer)));
        Hlock(handle(SintomasEnfermedad[I]));
        ReadByte(@SintomasEnfermedad[I]^^[1], sizeof(integer) *
NumSintomasEnfermedadPt^[I]);
        HUnlock(handle(SintomasEnfermedad[I]));
        TotalSize := OcurrenciasEnfermedad[I] * SizeOf(Probtype);
        NumberofParameters := NumberofParameters + OcurrenciasEnfermedad[I];
        Frecuencia[I] := FrecuenciaTypeHdl(NewHandle(TotalSize));
        Hlock(handle(Frecuencia[I]));
        ReadByte(@Frecuencia[I]^^[1], sizeof(ProbType) * OcurrenciasEnfermedad[I]);
        HUnlock(handle(Frecuencia[I]));
      END;
{writeln('Number of Parameters:=', NumberofParameters);}
    error := FSclose(refNum);
    ErrorDisk(error);
    IF false THEN
      BEGIN
```

```
{writeln(FileResul, 'Basepopulation=', Basepopulation, ' Numsintomas=', Numsintomas, '
NumeroEnfermedades=', NumeroEnfermedades);}
            FOR I := 1 TO 5 DO
               BEGIN
{writeln(FileResul, NombreEnfermedadPt^[I], ' Ocurrencias=', OcurrenciasEnfermedad[I]);}
                  FOR J := 1 TO NumSintomasEnfermedadPt^[I] DO
                     BEGIN
{write(FileResul, SintomasEnfermedad[I]^^[J], ProbSintoma[I]^^[J]);}
                     END;
{writeln;}
                  FOR J := 1 TO 10 DO
{write(FileResul, Frecuencia[I]^^[J]);}
{writeln(FileResul, '');}
               END;
            FOR J := 1 TO 3 DO
{writeln(FileResul, NombreSintomasPt^[J]);}
         END;
      inicializado := true;
      handlesdisposed := false;
      IF talkOn AND (idioma <> espanol) THEN
         say('The data has been readed');
      SetCursor(Arrow);
   END;

   PROCEDURE Actualizar;
      LABEL
         1;
      VAR
         pt : point;
         TypeList : SFTypeList;
         reply : SFReply;
         I, J : integer;
         reloj : cursHandle;
         Temp : Str255;
         VRefNum, Error, RefNum : integer;
         Dum : integer;
         TEXTR : RECT;

      PROCEDURE WriteByte (what : ptr;
                           num : longInt);
      BEGIN
         error := FSWrite(refNum, num, what);
         ErrorDisk(error);
      END;

   BEGIN
      SetPt(pt, 80, 80);
      GetIndString(temp, TxtStrID, 25);
      SFPutFile(pt, temp, '', NIL, reply);
      IF NOT reply.good THEN
         GOTO 1;
      Filename2 := reply.fName;
      VRefNum := reply.vRefNum;
      reloj := GetCursor(WatchCursor);
      SetCursor(reloj^^);
      IF FSOpen(Filename2, VRefNum, RefNum) = fnfErr THEN
         BEGIN
            Error := Create(Filename2, VRefNum, applType, DocType);
            ErrorDisk(error);
            error := FSOpen(Filename2, VRefNum, RefNum);
            ErrorDisk(error);
         END;
```

```
        WriteByte(@Basepopulation, sizeof(Basepopulation));
        writeByte(@Numsintomas, sizeof(NumSintomas));
        writeByte(@NumeroEnfermedades, sizeof(NumeroEnfermedades));
        FOR I := 1 TO NumeroEnfermedades DO
           BEGIN
              Hlock(handle(ProbSintoma[I]));
              WriteByte(@ProbSintoma[I]^^[1], Sizeof(ProbsintomaType));
              Hunlock(handle(ProbSintoma[I]));
           END;
        writeByte(@NombreSintomasPt^[1], sizeOf(nombreSintomasType) * NumSintomas);
        writeByte(@NombreEnfermedadPt^[1], sizeOf(NombreEnfermedadType) *
NumeroEnfermedades);
        writeByte(@NumSintomasEnfermedadPt^[1], sizeOf(NumSintomasEnfermedadType) *
NumeroEnfermedades);
        writeByte(@OcurrenciasEnfermedad[1], sizeof(OcurrenciasEnfermedadType) *
NumeroEnfermedades);
        FOR I := 1 TO NumeroEnfermedades DO
           BEGIN
              Hlock(handle(SintomasEnfermedad[I]));
              writeByte(@SintomasEnfermedad[I]^^[1], sizeof(integer) *
NumSintomasEnfermedadPt^[I]);
              HUnlock(handle(SintomasEnfermedad[I]));
              Hlock(handle(Frecuencia[I]));
              writeByte(@Frecuencia[I]^^[1], sizeof(ProbType) * OcurrenciasEnfermedad[I]);
              HUnlock(handle(Frecuencia[I]));
           END;
        error := FSclose(RefNum);
        ErrorDisk(error);
        error := FlushVol(NIL, VRefNum);
        ErrorDisk(error);
        IF true THEN
           BEGIN
{writeln(FileResul, 'Basepopulation=', Basepopulation, ' Numsintomas=', Numsintomas, '
NumeroEnfermedadees=', NumeroEnfermedades);}
              FOR I := 1 TO 5 DO
                 BEGIN
{writeln(FileResul, NombreEnfermedadPt^[I], ' Ocurrencias=', OcurrenciasEnfermedad[I]);}
                    FOR J := 1 TO NumSintomasEnfermedadPt^[I] DO
                       BEGIN
{write(FileResul, SintomasEnfermedad[I]^^[J], ProbSintoma[I]^^[J]);}
                       END;
                    writeln;
                    FOR J := 1 TO 10 DO
{write(FileResul, Frecuencia[I]^^[J]);}
{writeln(FileResul, '');}
                 END;
              FOR J := 1 TO 3 DO
{writeln(FileResul, NombreSintomasPt^[J]);}
           END;

1 :
        SetCursor(Arrow);
     END;
{-----------------------}
     PROCEDURE draw_sintomas;
        VAR
           i : integer;
     BEGIN
        FOR i := 1 TO NumSintomasEnfermedadPt^[currEnfermedad] DO
           BEGIN
              moveTo(1, i * 14);
              drawString(NombreSintomasPt^[SintomasEnfermedad[currEnfermedad]^^[i]]);
```

```
        END;
END;

PROCEDURE draw_enfermedad;
    VAR
        i, x, linea : integer;
BEGIN
    linea := 1;
    FOR i := 1 TO numeroEnfermedades DO
        FOR x := 1 TO NumSintomasEnfermedadPt^[i] DO
            IF sintomasEnfermedad[i]^^[x] = currSIntoma THEN
                BEGIN
                    moveTo(1, linea * 14);
                    linea := linea + 1;
                    drawString(NombreEnfermedadPt^[i]);
                END;
END;

PROCEDURE Information; {(N,Kopt : integer)}
    TYPE
        option = 0..2;
    VAR
        myEvent : EventRecord;
        WRecord : WindowRecord;
        TheWindow : WindowPtr;
        Columns, Lines : integer;
        LineWidth, ColumnWidth : integer;
        X1, Y1, X0, Y0, I, J, K, X2, Y2 : integer;
        KK : integer;
        X, Y, X4, Y3, X5, Y5, X6, Y6 : integer;
        Rectangle, rectangle1, rectangle2, rectangleOk, r : Rect;
        Kprev : integer;
        grafPort : GrafPtr;
        MousePoint : point;

    PROCEDURE draw_window;
        VAR
            i, k, j, desp : integer;
    BEGIN
        SetPort(theWindow);
        TextFace([bold]);
        TextMode(0);
        PenSize(2, 2);
        IF (N <= 100) THEN
            BEGIN
                LineWidth := 16;
                ColumnWidth := 32;
                desp := 2;
            END
        ELSE
            BEGIN
                LineWidth := 1600 DIV N;
                ColumnWidth := 32;
                desp := 1;
            END;
        Columns := 10;
        Lines := TRUNC((N - 1) / Columns) + 1;
        X4 := X0 + Columns * ColumnWidth;
        Y3 := Y0 + Lines * LineWidth + 10;
        SetRect(rectangleOk, 250, 5, 300, 25);
        PaintRoundRect(RectangleOk, 10, 10);
        InsetRect(rectangleOk, 2, 2);
```

```
        FillRoundRect(rectangleOk, 10, 10, White);
        getIndString(str, TxtStrID, 14);
        TextBox(pointer(ord(@Str) + 1), length(Str), rectangleOk, teJustCenter);
        TextSize(10);
        SetRect(Rectangle, X0, Y3, X4, Y3 + 50);
        FrameRect(rectangle);
        rectangle1 := rectangle;
        InsetRect(rectangle1, 2, 2);
        X1 := X0 + Columns * ColumnWidth;
        KK := 1;
        FOR I := 1 TO Lines + 1 DO
           BEGIN
              Y1 := Y0 + (I - 1) * LineWidth;
              MoveTo(X0, Y1);
              LineTo(X1, Y1);
              IF I <> 1 THEN
                 BEGIN
                    PenSize(1, 1);
                    TextFace([]);
                    FOR K := 1 TO Columns DO
                       BEGIN
                          X2 := X0 + (k - 1) * ColumnWidth + 4;
                          MoveTo(X2, Y1 - desp);
                          CASE kopt OF
                             1 :
                                BEGIN
                                   str := ' ';
                                   IF kk <= numeroenfermedades THEN
                                      Str := CONCAT(' ',
COPY(NombreEnfermedadPt^[EnfermedadOrden[KK]], 1, 1));
                                END;
                             2 :
                                BEGIN
                                   NumtoString(kk, Str);
                                END;
                             OTHERWISE
                          END;
                          DrawString(Str);

                          KK := KK + 1;
                       END;
                    PenSize(2, 2);
                 END;

           END;
        Y1 := Y0 + Lines * LineWidth;
        FOR I := 1 TO Columns + 1 DO
           BEGIN
              X1 := X0 + (I - 1) * ColumnWidth;
              MoveTo(X1, Y0);
              LineTo(X1, Y1);
           END;
        FOR I := 1 TO Lines DO
           BEGIN
              FOR J := 1 TO Columns DO
                 BEGIN
                    K := (I - 1) * Columns + J;
                    IF K > N THEN
                       BEGIN
                          X5 := X0 + (J - 1) * ColumnWidth + 1;
                          Y5 := Y0 + (I - 1) * LineWidth + 1;
                          SetRect(Rectangle2, X5, Y5, X5 + ColumnWidth, Y5 + LineWidth);
```

```
                        InsetRect(rectangle2, 1, 1);
                        FillRect(rectangle2, gray);
                    END;
                END;
            END;
        END;

    BEGIN
        X0 := 60;
        Y0 := 40;
        TheWindow := GetNewWindow(400, @WRecord, POINTER(-1));
        draw_window;
        Kprev := 0;
        REPEAT
            IF GetNextEvent(everyEvent, myEvent) THEN
                ;
            GetMouse(x, y);
            J := TRUNC((X - X0) / ColumnWidth) + 1;
            I := TRUNC((Y - Y0) / LineWidth) + 1;
            K := (I - 1) * Columns + J;
            IF (Y > Y0) AND (I <= Lines) AND (X > X0) AND (J <= Columns) AND (K <= N) THEN
                BEGIN
                    IF button THEN
                        BEGIN
                            X5 := X0 + (J - 1) * ColumnWidth + 1;
                            Y5 := Y0 + (I - 1) * LineWidth + 1;
{-------------------------}
                            IF kopt = 1 THEN
                                BEGIN
                                    WITH screenBits.bounds DO
                                        setRect(r, left + 10, 40, right - 10, bottom - 50);
                                    currEnfermedad := EnfermedadOrden[K];
                                    display_list(r, NombreEnfermedadPt^[EnfermedadOrden[K]],
NumSintomasEnfermedadPt^[EnfermedadOrden[K]], 14, @draw_sintomas);
                                    draw_window;
                                    clipRect(theWindow^.portRect);
                                END
                            ELSE IF kopt = 2 THEN
                                BEGIN
                                    WITH screenBits.bounds DO
                                        setRect(r, 150, 50, 360, bottom - 20);
                                    currSintoma := k;
                                    numEnferSinto := 0;
                                    FOR i := 1 TO numeroEnfermedades DO
                                        FOR x := 1 TO NumSintomasEnfermedadPt^[i] DO
                                            IF sintomasEnfermedad[i]^^[x] = k THEN
                                                numEnferSinto := NumEnferSinto + 1;
                                    display_list(r, NombreSintomasPt^[k], NumEnferSinto, 14,
@draw_enfermedad);
                                    draw_window;
                                    clipRect(theWindow^.portRect);
                                END;
                            SetRect(Rectangle2, X5, Y5, X5 + ColumnWidth, Y5 + LineWidth);
                            InsetRect(rectangle2, 1, 1);
                            X6 := X0 + (J - 1) * ColumnWidth + 4;
                            Y6 := Y0 + I * LineWidth - 2;
                            REPEAT
                            UNTIL NOT button;
                        END;
                    IF Kprev <> K THEN
                        BEGIN
                            Kprev := K;
```

```
                    EraseRect(rectangle1);
                    Str := '  ';
                    CASE Kopt OF
                       1 :
                          BEGIN
                             Str := NombreEnfermedadPt^[EnfermedadOrden[K]];
                             FOR KK := 1 TO NumSintomasEnfermedadPt^[EnfermedadOrden[K]]
DO
                                Str := Concat(Str,
StringOf(SintomasEnfermedad[EnfermedadOrden[K]]^^[KK]));
                          END;
                       2 :
                          Str := NombreSintomasPt^[K];
                    END;
                    TextBox(pointer(ord(@Str) + 1), length(Str), rectangle1, teJustLeft);
                 END;
              END
           ELSE
              BEGIN
                 K := 0;
                 IF Kprev <> K THEN
                    BEGIN
                       Kprev := K;
                       Str := '  ';
                       EraseRect(rectangle1);
                       TextBox(pointer(ord(@Str) + 1), length(Str), rectangle1, teJustCenter);
                    END;
              END;
           MousePoint.h := X;
           MousePoint.v := Y;
        UNTIL button AND PtinRect(MousePoint, rectangleOK);
        DisposeWindow(TheWindow);
     END;

     PROCEDURE Sintomas; {(N : integer)}
        CONST
           RectHeight = 50;

        VAR
           myEvent : eventRecord;
           WRecord : WindowRecord;
           TheWindow : WindowPtr;
           Columns, Lines : integer;
           LineWidth, ColumnWidth, Disp : integer;
           X1, Y1, X0, Y0, I, J, K, X2, Y2 : integer;
           KK : integer;
           X, Y, X4, Y3, X5, Y5, X6, Y6 : integer;
           Rectangle, rectangle1, rectangle2, rectangleOk, r : Rect;
           Kprev : integer;
           grafPort : GrafPtr;
           MousePoint : point;
           keys : keymap;

        PROCEDURE draw_window;
           VAR
              i, k, j : integer;
        BEGIN
           X0 := 60;
           Y0 := 8;
           SetPort(theWindow);
           TextFace([bold]);
           TextMode(0);
```

```
PenSize(2, 2);
IF (N <= 100) THEN
   BEGIN
      LineWidth := 16;
      ColumnWidth := 32
   END
ELSE
   BEGIN
      LineWidth := 1600 DIV N;
      ColumnWidth := 32
   END;
Columns := 10;
Lines := TRUNC((N - 1) / Columns) + 1;
X4 := X0 + Columns * ColumnWidth;
Y3 := Y0 + Lines * LineWidth + 10;
Disp := Trunc((X4 - X0 - ColumnWidth) / 3.5);
SetRect(Rectangle1, 0, 0, ColumnWidth, LineWidth);
OffsetRect(Rectangle1, X0 DIV 3, thewindow^.portrect.bottom - 30);
FrameRect(Rectangle1);
Rectangle2 := Rectangle1;
InsetRect(rectangle2, 2, 2);
FillRect(Rectangle2, Black);
Rectangle2 := Rectangle1;
OffsetRect(Rectangle2, 0, LineWidth);
getIndString(str, TxtStrID, 10);
TextBox(pointer(ord(@Str) + 1), length(Str), rectangle2, teJustCenter);
OffsetRect(Rectangle1, Disp, 0);
FrameRect(Rectangle1);
Rectangle2 := Rectangle1;
InsetRect(rectangle2, 2, 2);
FillRect(Rectangle2, gray);
Rectangle2 := Rectangle1;
OffsetRect(Rectangle2, 0, LineWidth);
getIndString(str, TxtStrID, 11);
TextBox(pointer(ord(@Str) + 1), length(Str), rectangle2, teJustCenter);
OffsetRect(Rectangle1, Disp, 0);
FrameRect(Rectangle1);
Rectangle2 := Rectangle1;
InsetRect(rectangle2, 2, 2);
FillRect(Rectangle2, ltgray);
Rectangle2 := Rectangle1;
OffsetRect(Rectangle2, 0, LineWidth);
InsetRect(rectangle2, -30, 0);
getIndString(str, TxtStrID, 12);
TextBox(pointer(ord(@Str) + 1), length(Str), rectangle2, teJustCenter);
OffsetRect(Rectangle1, Disp, 0);
FrameRect(Rectangle1);
Rectangle2 := Rectangle1;
OffsetRect(Rectangle2, 0, LineWidth);
InsetRect(rectangle2, -30, 0);
getIndString(str, TxtStrID, 13);
TextBox(pointer(ord(@Str) + 1), length(Str), rectangle2, teJustCenter);
RectangleOK := Rectangle1;
OffSetRect(RectangleOK, Disp, 0);
RectangleOk.right := RectangleOk.left + 60;
RectangleOk.top := (RectangleOk.top + RectangleOk.bottom) DIV 2 - 10;
RectangleOk.bottom := RectangleOk.top + 20;
getIndString(str, TxtStrID, 14);
TextBox(pointer(ord(@Str) + 1), length(Str), rectangleOk, teJustCenter);
FrameRoundRect(RectangleOk, 20, 20);
TextSize(10);
SetRect(Rectangle, X0, Y3, X4, Y3 + RectHeight);
```

```
  FrameRect(rectangle);
  rectangle1 := rectangle;
  InsetRect(rectangle1, 2, 2);
  X1 := X0 + Columns * ColumnWidth;
  KK := 1;
  FOR I := 1 TO Lines + 1 DO
     BEGIN
        Y1 := Y0 + (I - 1) * LineWidth;
        MoveTo(X0, Y1);
        LineTo(X1, Y1);
        IF I <> 1 THEN
           BEGIN
              PenSize(1, 1);
              TextFace([]);
              FOR K := 1 TO Columns DO
                 BEGIN
                    X2 := X0 + (k - 1) * ColumnWidth + 4;
                    X5 := X2 - 3;
                    Y5 := Y1 + 1 - LineWidth;
                    SetRect(Rectangle2, X5, Y5, X5 + ColumnWidth, Y5 + LineWidth);
                    InsetRect(Rectangle2, 1, 1);
                    CASE SintomaEnfermoClass[KK] OF
                       1 :
                          FillRect(rectangle2, black);
                       2 :
                          FillRect(rectangle2, gray);
                       3 :
                          FillRect(rectangle2, ltgray);
                       OTHERWISE
                    END;
                    MoveTo(X2, Y1 - 2);
                    NumTOString(KK, Str);
                    DrawString(Str);
                    KK := KK + 1;
                 END;
              PenSize(2, 2);
           END;
     END;
  Y1 := Y0 + Lines * LineWidth;
  FOR I := 1 TO Columns + 1 DO
     BEGIN
        X1 := X0 + (I - 1) * ColumnWidth;
        MoveTo(X1, Y0);
        LineTo(X1, Y1);
     END;
  FOR I := 1 TO Lines DO
     BEGIN
        FOR J := 1 TO Columns DO
           BEGIN
              K := (I - 1) * Columns + J;
              IF K > N THEN
                 BEGIN
                    X5 := X0 + (J - 1) * ColumnWidth + 1;
                    Y5 := Y0 + (I - 1) * LineWidth + 1;
                    SetRect(Rectangle2, X5, Y5, X5 + ColumnWidth, Y5 + LineWidth);
                    InsetRect(rectangle2, 1, 1);
                    FillRect(rectangle2, ltgray);
                 END;
           END;
     END;
END;
```

```
BEGIN
   TheWindow := GetNewWindow(400, @WRecord, POINTER(-1));
   draw_window;
   Kprev := 0;
   REPEAT
      IF GetNextEvent(EveryEvent, myEvent) THEN
         ;
      GetMouse(x, y);
      J := TRUNC((X - X0) / ColumnWidth) + 1;
      I := TRUNC((Y - Y0) / LineWidth) + 1;
      K := (I - 1) * Columns + J;
      IF (Y > Y0) AND (I <= Lines) AND (X > X0) AND (J <= Columns) AND (K <= N) THEN
         BEGIN
            IF button THEN
               BEGIN
                  getKeys(keys);
                  IF bitTst(@keys, 61) THEN
                     BEGIN
                        WITH screenBits.bounds DO
                           setRect(r, 150, 50, 360, bottom - 20);
                        currSintoma := k;
                        numEnferSinto := 0;
                        FOR i := 1 TO numeroEnfermedades DO
                           FOR x := 1 TO NumSintomasEnfermedadPt^[i] DO
                              IF sintomasEnfermedad[i]^^[x] = k THEN
                                 numEnferSinto := NumEnferSinto + 1;
                        display_list(r, NombreSintomasPt^[k], NumEnferSinto, 14,
@draw_enfermedad);
                        draw_window;
                        kprev := 0;{para que redibuje el nombre del sintoma}
                     END
                  ELSE
                     BEGIN
                        X5 := X0 + (J - 1) * ColumnWidth + 1;
                        Y5 := Y0 + (I - 1) * LineWidth + 1;
                        SetRect(Rectangle2, X5, Y5, X5 + ColumnWidth, Y5 + LineWidth);
                        InsetRect(rectangle2, 1, 1);
                        Sintomaenfermoclass[K] := (Sintomaenfermoclass[K] + 1) MOD 4;
                        X6 := X0 + (J - 1) * ColumnWidth + 4;
                        Y6 := Y0 + I * LineWidth - 2;
                        CASE Sintomaenfermoclass[K] OF
                           0 :
                              BEGIN
                                 FillRect(rectangle2, white);
                                 MoveTo(X6, Y6);
                                 NumTOString(K, Str);
                                 DrawString(Str);
                              END;
                           1 :
                              BEGIN
                                 FillRect(rectangle2, black);
                                 TextMode(0);
                                 MoveTo(X6, Y6);
                                 NumTOString(K, Str);
                                 DrawString(Str);
                                 PenPat(black);
                              END;
                           2 :
                              BEGIN
                                 FillRect(rectangle2, gray);
                                 PenPat(White);
                                 MoveTo(X6, Y6);
```

```
                                    NumTOString(K, Str);
                                    DrawString(Str);
                                    PenPat(black);
                                END;
                            3 :
                                BEGIN
                                    FillRect(rectangle2, ltgray);
                                    PenPat(White);
                                    MoveTo(X6, Y6);
                                    NumTOString(K, Str);
                                    DrawString(Str);
                                    PenPat(black);
                                END;
                        END;
                        REPEAT
                        UNTIL NOT button;
                    END;
                IF Kprev <> K THEN
                    BEGIN
                        Kprev := K;
                        EraseRect(rectangle1);
                        TextBox(pointer(ord(@NombreSintomasPt^[K]) + 1),
length(NombreSintomasPt^[K]), rectangle1, teJustCenter);
                    END;
            END;
        IF Kprev <> K THEN
            BEGIN
                Kprev := K;
                EraseRect(rectangle1);
                TextBox(pointer(ord(@NombreSintomasPt^[K]) + 1),
length(NombreSintomasPt^[K]), rectangle1, teJustCenter);
            END;
    END;
    MousePoint.h := X;
    MousePoint.v := Y;
UNTIL button AND PtinRect(MousePoint, rectangleOK);
NumSintomasEnfermo := 0;
FOR I := 1 TO N DO
    BEGIN
        CASE SintomaEnfermoClass[I] OF
            1, 2, 3 :
                BEGIN
                    NumSintomasEnfermo := NumSintomasEnfermo + 1;
                    SintomaEnfermo[NumSintomasEnfermo] := I;
                    SintomaEnfermoDef[NumSintomasEnfermo] := SintomaEnfermoClass[I];
                END;
            OTHERWISE
        END;
    END;
{   writeln('Numsintomasenfermo = ', Numsintomasenfermo);}
{for KK := 1 to Numsintomasenfermo do}
{write(SintomaEnfermo[kk], SintomaEnfermoDef[KK]);}
{writeln(' ');}
    DisposeWindow(TheWindow);
END;

PROCEDURE Ordenar;
    VAR
        B : integer;
        I, K : integer;
        P : real;
BEGIN
```

```
    J := N - 1;
    REPEAT
      K := 0;
      FOR I := 1 TO J DO
        IF Sintomas[I] > Sintomas[I + 1] THEN
          BEGIN
            B := Sintomas[I];
            Sintomas[I] := Sintomas[I + 1];
            Sintomas[I + 1] := B;
            P := Pr[I];
            Pr[I] := Pr[I + 1];
            Pr[I + 1] := P;
            K := 1;
            J := I;
          END;
    UNTIL K = 0;
  END;

  PROCEDURE Ordenar2 (N, M : integer;
                      VAR dialogpointer : dialogPtr);
    VAR
      Max : real;
      I, J : integer;
      C : integer;
      B : real;

    PROCEDURE DialogProb (DialogID : integer;
                          VAR dialogpointer : dialogPtr);
      VAR
        DItem, CType : integer;
        CHdl : Handle;
        CRect : Rect;
        I, Lstr, dum : INTEGER;
    BEGIN
      IF EndDiagnostico = True THEN
        BEGIN
          dialogpointer := GetNewDialog(DialogID, NIL, POINTER(-1));
          IF OptionVerosimilitud = true THEN
            getIndString(str, TxtStrID, 3);
          IF Optionprobabilidad = true THEN
            getIndString(str, TxtStrID, 2);
          GetDItem(dialogpointer, TitleProbabilidadDialogId, CType, CHdl, CRect);
          SetItext(CHdl, Str);
          EndDiagnostico := False;
        END;
      IF DialogID = ProbabilidadDialogID THEN
        set_dialog_text(dialogpointer, 14, 30);{cancelar}
      FOR I := 1 TO M DO
        BEGIN
          GetDItem(dialogpointer, I + 1, CType, CHdl, CRect);
          IF Prob[I] > 0.001 THEN
            Str := stringof(NombreEnfermedadPt^[Nu[I]])
          ELSE
            Str := '                    ';
          SetItext(CHdl, Str);
          GetDItem(dialogpointer, I + 6, CType, CHdl, CRect);
          IF Prob[I] > 0.001 THEN
            Str := stringof(Prob[I] : 8 : 3)
          ELSE
            Str := '      ';
{WRITELN(STR, ' len=', length(Str));}
          SetItext(CHdl, Str);
```

```
            END;
        REPEAT
            IF ((Prob[1] >= 0.999) OR (NumsintomasEnfermo = Numsintomas)) THEN
                BEGIN
                    EnableItem(dialogpointer, 1, false);
                    getIndString(str, TxtStrID, 8);
                    ParamText('', Str, '', '');
                    Dum := NoteAlert(NoteAlertID, NIL);
                END;
            ModalDialog(NIL, dItem);
            IF (ditem = 14) THEN
                BEGIN
                    cancel := True;
                    Enddiagnostico := True;
                END;
        UNTIL (ditem = 1) OR (ditem = 14);
    END;

BEGIN
    IF M > N THEN
        M := N;
    FOR I := 1 TO M DO
        BEGIN
            Max := Prob[I];
            K := I;
            FOR J := I + 1 TO N DO
                IF Prob[J] > Max THEN
                    BEGIN
                        Max := Prob[J];
                        K := J;
                    END;
            B := Prob[I];
            Prob[I] := Prob[K];
            Prob[K] := B;
            C := Nu[I];
            Nu[I] := Nu[K];
            Nu[K] := C;
        END;
    DialogProb(ProbabilidadDialogId, dialogpointer);
END;

PROCEDURE bitsdesc;
    VAR
        L1 : integer;
BEGIN
    FOR L1 := 0 TO N - 1 DO
        IF BitTst(@L, 31 - L1) THEN
            write('1')
        ELSE
            write('0');
END;

FUNCTION SintomaEnfermedad (VAR I, J : integer) : integer;
BEGIN
    SintomaEnfermedad := SintomasEnfermedad[J]^^[I];
END;

FUNCTION FrecuenciaCasoEnfermedad (VAR I, J : integer) : real;
BEGIN
    FrecuenciaCasoEnfermedad := Frecuencia[J]^^[I];
END;
```

```
    PROCEDURE Diagnostico;
      VAR
        I, J, L1, K3 : integer;
        RRect : rect;
        reloj : cursHandle;
        S1 : ProbType;
        Dum : integer;
        S : extended;
        SintomaContraste : SintomasEnfermoType;
        SintomaContrasteDef : SintomaEnfermoDefType;
    BEGIN
      diagnosticofailed := false;
      reloj := GetCursor(WatchCursor);
      SetCursor(reloj^^);
      SetRect(RRect, 20, 80, 510, 200);
      SetTextRect(RRect);
{ShowText;}
      Ordenar1(SintomaEnfermo, SintomaEnfermoDef, NumSintomasEnfermo);
      FOR I := 1 TO NumeroEnfermedades DO
        BEGIN
          S1 := 0;
          Hlock(handle(Frecuencia[I]));
          FrecuenciaPtr := Frecuencia[I]^;
          FOR J := 1 TO NumSintomasEnfermo DO
            BEGIN
              SintomaContraste[J] := SintomaEnfermo[J];
              SintomaContrasteDef[J] := SintomaEnfermoDef[J];
            END;
          SintomasComunes(I, NumsintomasEnfermo, SintomaEnfermo, SintomaEnfermoDef);
{writeln(' ');}
{writeln('Enfermedad='; I : 3, ' Numsintomascomunes=', NumSintomascomunes : 2, '
Numsintomasnocomunes=', Numsintomasnocomunes : 2);}
          IF NumSintomasComunes <> 0 THEN
            BEGIN
              K1 := 0;
              K2 := 0;
              FOR L1 := 1 TO NumSintomasComunes DO
                BEGIN
                  IF SintomaComunDef[L1] <> 3 THEN
                    Bitset(@K1, 32 - SintomaComun[L1]);
                  IF SintomaComunDef[L1] = 1 THEN
                    Bitset(@K2, 32 - SintomaComun[L1]);
                END;
{   bitsdesc(K1, NumSintomasEnfermedadPt^[I]);}
{   writeln(' =  K1');}
{   bitsdesc(K2, NumSintomasEnfermedadPt^[I]);}
{   writeln(' =  K2');}
              FOR K3 := 0 TO OcurrenciasEnfermedad[I] - 1 DO
                BEGIN
{bitsdesc(K3, NumSintomasEnfermedadPt^[I]);}
{bitsdesc(K3, NumSintomasEnfermedadPt^[I]);}
                  IF BitAnd(K1, K3) = K2 THEN
                    BEGIN
                      S1 := S1 + FrecuenciaPtr^[K3 + 1]
{writeln('  S1=', S1);}
                    END;
{writeln('   PE', Trunc(Penfermedad[I]));}
                END;
            END
          ELSE
            BEGIN
              FOR K3 := 0 TO OcurrenciasEnfermedad[I] - 1 DO
```

```
                    BEGIN
                        S1 := S1 + FrecuenciaPtr^[K3 + 1]
                    END;
                END;
            PEnfermedad[I] := S1;
{writeln('  PEnfermedad[', I : 3, ']=', Trunc(Penfermedad[I]));}
            IF PEnfermedad[I] <> 0.0 THEN
                BEGIN
                    S := 1.0;
                    FOR J := 1 TO NumSintomasNoComunes DO
                        BEGIN
                            Hlock(handle(ProbSintoma[I]));
                            CASE SintomaNoComunDef[J] OF
                                1 :
                                    IF OptionProbSintomasNoRelevantes THEN
                                        S := 0.0
                                    ELSE
                                        S := S * ProbSintoma[I]^^[SintomaNoComun[J]];
                                2 :
                                    IF OptionProbSintomasNoRelevantes THEN
                                        S := S
                                    ELSE
                                        S := S * (1.0 - ProbSintoma[I]^^[SintomaNoComun[J]]);
                                OTHERWISE
                            END;
                            Hunlock(handle(ProbSintoma[I]));
                        END;
                    prob[I] := S * PEnfermedad[I];
{writeln('S, PEnfermedad[I],S1=', S : 12 : 8, PEnfermedad[I], S1);}
                END
            ELSE
                Prob[I] := 0.0;
            HUnlock(handle(Frecuencia[I]));
            IF Prob[I] <> 0.0 THEN
{writeln('Probabilidad enfermedad ', I, ' = ', Prob[I] : 8 : 5);}
        END;
    S := 0.0;
    IF OptionProbabilidad = True THEN
        FOR I := 1 TO NumeroEnfermedades DO
            S := S + Prob[I];
    IF OptionVerosimilitud = True THEN
        FOR I := 1 TO NumeroEnfermedades DO
            BEGIN
                IF S < Prob[I] THEN
                    S := Prob[I];
            END;
    IF S <= 0.0 THEN
        BEGIN
            getIndString(str1, TxtStrID, 21);
            ParamText('', str1, '', '');
            Dum := NoteAlert(NoteAlertID, NIL);
            Diagnosticofailed := true;
        END;
    FOR I := 1 TO NumeroEnfermedades DO
        IF diagnosticofailed THEN
            Prob[I] := 0.0
        ELSE
            Prob[I] := Prob[I] / S;
{writeln(I, Prob[I], S);}
    SetCursor(Arrow);
END;
```

```
PROCEDURE orden;
   VAR
      I : integer;
BEGIN
   FOR I := 1 TO NumeroEnfermedades DO
      Nu[I] := I;
   ordenar2(NumeroEnfermedades, NListEnfermedades, dialogpointer);
END;

PROCEDURE EnfermoaBase;
   VAR
      J, I, J1 : integer;
      L : longint;
      Dum : integer;
      done : boolean;
      L1, K3, ditem : integer;
      EI : Probtype;
      Re : real;
      Added : boolean;
      DialogP : DialogPtr;
      CHdl : Handle;
      CRect : Rect;
      cType : integer;

BEGIN
   added := false;
   done := False;
   getIndString(str1, TxtStrID, 7);
   Str2 := '';
   DialogP := GetNewDialog(FiledialogID, NIL, pointer(-1));
   GetDItem(dialogP, 3, cType, chdl, Crect);
   SetItext(chdl, str1);
   REPEAT
      Modaldialog(NIL, DItem);
      CASE ditem OF
         2 :
            done := true;
         1 :
            BEGIN
               trim(Str2);
               UprString(Str2, True);
               I := 0;
               REPEAT
                  I := I + 1;
                  Str3 := NombreEnfermedadPt^[I];
                  UprString(Str3, True);
               UNTIL (Str3 = Str2) OR (I >= NumeroEnfermedades);
               IF Str3 = Str2 THEN
                  BEGIN
                     IF Numsintomasenfermo <> 0 THEN
                        BEGIN
                           IF NOT added THEN
                              BEGIN
                                 Basepopulation := Basepopulation + Factorpopulation;
                                 added := true;
                              END;
                           Hlock(handle(Frecuencia[I]));
                           EI := 0;
                           FOR J := 1 TO OcurrenciasEnfermedad[I] DO
                              EI := EI + Frecuencia[I]^^[J];
                           Hunlock(handle(Frecuencia[I]));
                           Hlock(handle(ProbSintoma[I]));
```

```
                                FOR J := 1 TO Numsintomasenfermo DO
                                   BEGIN
                                      CASE SintomaEnfermoDef[I] OF
                                         1 :
                                            ProbSintoma[I]^^[SintomaEnfermo[J]] :=
(ProbSintoma[I]^^[SintomaEnfermo[J]] * EI / Factorpopulation + Factorpopulation) / (EI DIV
Factorpopulation + 1);
                                         2 :
                                            ProbSintoma[I]^^[SintomaEnfermo[J]] :=
(ProbSintoma[I]^^[SintomaEnfermo[J]] * EI / Factorpopulation) / (EI DIV Factorpopulation + 1);
                                         OTHERWISE
                                      END;
                                   END;
                                Hunlock(handle(ProbSintoma[I]));
                                SintomasComunes(I, NumsintomasEnfermo, SintomaEnfermo,
SintomaEnfermoDef);
                                IF NumsintomasComunes <> 0 THEN
                                   BEGIN
                                      EI := 0;
                                      K1 := 0;
                                      K2 := 0;
                                      FOR L1 := 1 TO NumSintomasComunes DO
                                         BEGIN
                                            IF SintomaComunDef[L1] <> 3 THEN
                                               Bitset(@K1, 32 - SintomaComun[L1]);
                                            IF SintomaComunDef[L1] = 1 THEN
                                               Bitset(@K2, 32 - SintomaComun[L1]);
                                         END;
                                      FOR J := 1 TO OcurrenciasEnfermedad[I] DO
                                         BEGIN
                                            J1 := J - 1;
{   bitsdesc(K1, NumSintomasEnfermedad[I]);}
{   writeln(' = K1');}
{   bitsdesc(K2, NumSintomasEnfermedad[I]);}
{   writeln(' = K2');}
                                            IF Bitand(K1, J1) = Bitand(K1, K2) THEN
                                               EI := EI + Frecuencia[I]^^[J]
                                         END;
                                      Re := Factorpopulation / EI;
                                      K1 := 0;
                                      K2 := 0;
                                      FOR L1 := 1 TO NumSintomasComunes DO
                                         BEGIN
                                            IF SintomaComunDef[L1] <> 3 THEN
                                               Bitset(@K1, 32 - SintomaComun[L1]);
                                            IF SintomaComunDef[L1] = 1 THEN
                                               Bitset(@K2, 32 - SintomaComun[L1]);
                                         END;
                                      FOR J := 1 TO OcurrenciasEnfermedad[I] DO
                                         BEGIN
                                            J1 := J - 1;
{   bitsdesc(K1, NumSintomasEnfermedad[I]);}
{   writeln(' = K1');}
{   bitsdesc(K2, NumSintomasEnfermedad[I]);}
{   writeln(' = K2');}
                                            IF Bitand(K1, J1) = Bitand(K1, K2) THEN
                                               Frecuencia[I]^^[J] := TRUNC(Frecuencia[I]^^[J] * (1 +
Re));
                                         END;
                                   END;
                                done := true;
                             END
```

```
                        ELSE
                            BEGIN
                                done := true;
                                getIndString(str, TxtStrID, 22);
                                ParamText('', Str, '', '');
                                Dum := NoteAlert(NoteAlertID, NIL);
                            END;
                    END
                ELSE
                    BEGIN
                        getIndString(str1, TxtStrID, 23);
                        Str := Stringof(str1, Str2);
                        ParamText('', Str, '', '');
                        Dum := NoteAlert(NoteAlertID, NIL);
                    END;
            END;
        4 :
            BEGIN
                GetDItem(dialogP, 4, ctype, chdl, crect);
                GetItext(chdl, str2);
            END;
        OTHERWISE
    END;
  UNTIL done;
  Disposdialog(dialogp);
  writeln('----------');
{escribir;}
END;

PROCEDURE SimularEnfermo;
  VAR
    K : integer;
    Nenf, Nstop : Longint;
    J : Longint;

  FUNCTION Ranf : real;
    CONST
      Fact = 0.00001525925474;
    VAR
      B : LONGINT;
  BEGIN
    B := random;
    B := B + 32767;
    Ranf := B * Fact;
  END;

BEGIN
  Randseed := Tickcount;
  Enfermedadsimulada := Trunc(Ranf * NumeroEnfermedades) + 1;
  Nenf := 0;
  FOR J := 1 TO OcurrenciasEnfermedad[Enfermedadsimulada] DO
    Nenf := Nenf + Frecuencia[Enfermedadsimulada]^^[J];
  Nstop := Trunc(Nenf * ranf);
  J := 0;
  Nenf := 0;
  REPEAT
    J := J + 1;
    Nenf := Nenf + Frecuencia[Enfermedadsimulada]^^[J];
  UNTIL Nenf >= Nstop;
  J := J - 1;
  NumsintomasEnfermo := NumSintomasEnfermedadPt^[Enfermedadsimulada];
  FOR K := 1 TO NumSintomasEnfermedadPt^[Enfermedadsimulada] DO
```

```
        BEGIN
            SintomaEnfermo[K] := SintomasEnfermedad[Enfermedadsimulada]^^[K];
            IF BitTst(@J, 32 - K) THEN
                BEGIN
                    SintomaEnfermoDef[K] := 1;
                    SintomaEnfermoClass[SintomaEnfermo[K]] := 1;
                END
            ELSE
                BEGIN
                    IF OptionSimulation THEN
                        BEGIN
                            SintomaEnfermoDef[K] := 1;
                            SintomaEnfermoClass[SintomaEnfermo[K]] := 1;
                        END
                    ELSE
                        BEGIN
                            SintomaEnfermoDef[K] := 2;
                            SintomaEnfermoClass[SintomaEnfermo[K]] := 2;
                        END;
                END;
        END;
    END;

    PROCEDURE MostrarEnfermedad;

        VAR
            Dum : integer;
    BEGIN
        getIndString(str1, TxtStrID, 1);
        Str := Stringof(str1, NombreEnfermedadPt^[Enfermedadsimulada], ' (',
OrdenEnfermedad[Enfermedadsimulada] : 3, ')');
        ParamText('', Str, '', '');
        Dum := NoteAlert(NoteAlertID, NIL);
    END;
END.
```

Unidad INICIALIZACIÓN

```
UNIT inicializacion;

INTERFACE
   USES
      SkelGlobals, experto;

   PROCEDURE inicializar;

IMPLEMENTATION

   PROCEDURE inicializar;
      LABEL
         1;
      VAR
         typeList : SFTypeList;
         reply : SFReply;
         pt : point;
         i, j, l1, k1 : integer;
         reloj : cursHandle;
         L : LongInt;
         S, S1, pro : extended;
         CH1dl, CH2dl : Handle;
         C1Type, C2Type : integer;
         C1Rect, C2Rect : rect;
         DialogP : DialogPtr;
         dItem : integer;
         WRecord : WindowRecord;
         TheWindow : WindowPtr;
         Rectangle1, rectangle2 : rect;
         Dum : integer;
         FIJ : ARRAY[1..NmaxSintomas] OF extended;

      PROCEDURE Escritura;
         VAR
            I, J : integer;
      BEGIN
         writeln('Basepopulation = ', Basepopulation);
         FOR I := 1 TO Numeroenfermedades DO
            BEGIN
               writeln('Enfermedad : ', NombreEnfermedadPt^[I]);
               FOR J := 1 TO numsintomasenfermedadPt^[I] DO
                  BEGIN
                     write(Sintomasenfermedad[I]^^[J] : 3);
                  END;
               writeln(' ');
               FOR J := 1 TO ocurrenciasEnfermedad[I] DO
                  BEGIN
                     bitsdesc(J - 1, NumsintomasEnfermedadPt^[I]);
                     writeln('.....', Frecuencia[I]^^[J]);
                  END;
            END;
         FOR I := 1 TO Numeroenfermedades DO
            BEGIN
               FOR J := 1 TO Numsintomas DO
                  BEGIN
                     writeln('prosintoma[', J : 3, ']=', Probsintoma[I]^^[J]);
                  END;
            END;
```

```
    END;

  BEGIN
    SetPt(pt, 80, 60);
    typeList[0] := 'TEXT';
    SFGetFile(pt, '', NIL, 1, typeList, NIL, reply);
    IF NOT reply.good THEN
      GOTO 1;
    Filename1 := reply.fName;

    inicializado := true;

    SetRect(rectangle1, 10, 10, 250, 50);
    SetRect(rectangle2, 10, 60, 250, 140);
    I := 1;

    reloj := GetCursor(WatchCursor);
    SetCursor(reloj^^);
    SetRect(RRect, 20, 80, 510, 200);
    S1 := 0.0;
    reset(ff, Filename1);
    readln(ff, Basepopulation);
    readln(ff, NombreEnfermedadPt^[I]);
    WHILE NombreEnfermedadPt^[I] <> '99999' DO
      BEGIN
        Read(ff, Pro);
        IF (pro > 1.0) OR (pro < 0.0) THEN
          BEGIN
            getIndString(str1, TxtStrID, 15);
            getIndString(str2, TxtStrID, 16);
            getIndString(str3, TxtStrID, 17);
            Str := stringof(str1, NombreEnfermedadPt^[I], Str2, pro, Str3);
            ParamText('', Str, '', '');
            Dum := NoteAlert(NoteAlertID, NIL);
          END;
        S1 := S1 + pro;
        J := 1;
        Read(ff, Sintoma[J]);
        WHILE Sintoma[J] <> 999 DO
          BEGIN
            Read(ff, Pr[J]);
            IF (Pr[J] > 1.0) OR (Pr[J] < 0.0) THEN
              BEGIN
                Str := stringof(Str1, NombreEnfermedadPt^[I], str2, Pr[J], Str3);
                ParamText('', Str, '', '');
                Dum := NoteAlert(NoteAlertID, NIL);
              END;
            Read(ff, Sintoma[J]);
          END;
        I := I + 1;
        readln(ff);
        readln(ff, NombreEnfermedadPt^[I]);
      END;
     read(ff, k);
     Numsintomas := 0;
     WHILE k <> 999 DO
      BEGIN
        readln(ff, nombreSintomasPt^[k]);
{  writeln( K, ' ', nombreSintomasPt^[k]); }
        read(ff, k);
        NumSintomas := Numsintomas + 1;
      END;
```

```
        FOR j := 1 TO Numsintomas DO
            FIJ[j] := 0.0;
{writeln('S1=', S1);}
        IF (S1 < -0.001) OR (S1 > 1.001) THEN
            BEGIN
                getIndString(str1, TxtStrID, 18);
                ParamText('', Str1, '', '');
                Dum := NoteAlert(NoteAlertID, NIL);
            END;
        close(ff);
        NumeroEnfermedades := I - 1;
        I := 1;
        reset(ff, Filename1);
        readln(ff, Basepopulation);
        readln(ff, NombreEnfermedadPt^[I]);
        trim(NombreEnfermedadPt^[I]);
        WHILE NombreEnfermedadPt^[I] <> '99999' DO
            BEGIN
                Read(ff, Pro);
                J := 1;
                Read(ff, Sintoma[J]);
{   write(Sintoma[J] : 4);}
                WHILE Sintoma[J] <> 999 DO
                    BEGIN
                        Read(ff, Pr[J]);
{   write(Pr[J] : 5 : 2);}
                        FIJ[Sintoma[J]] := FIJ[Sintoma[J]] + Pr[J] * Pro / S1;
                        J := J + 1;
                        Read(ff, Sintoma[J]);
{   write(Sintoma[J] : 4);}
                    END;
{   writeln(' ');}
                NumSintomasEnfermedadPt^[I] := J - 1;
                Ordenar(sintoma, Pr, NumSintomasEnfermedadPt^[I]);
{   write('Numero sintomas =', NumSintomasEnfermedadPt^[I], ' : ');}
                SintomasEnfermedad[I] := SintomaHdl(NewHandle(NMaxSintomasporEnfermedad *
Sizeof(Integer)));
                Hlock(handle(SintomasEnfermedad[I]));
                FOR J := 1 TO NumSintomasEnfermedadPt^[I] DO
                    BEGIN
                        SintomasEnfermedad[I]^^[J] := Sintoma[J];
{write(Sintoma[J] : 4);}
                    END;
{   writeln(' ');}
                HUnlock(handle(SintomasEnfermedad[I]));
                I := I + 1;
                readln(ff);
                readln(ff, NombreEnfermedadPt^[I]);
                trim(NombreEnfermedadPt^[I]);
{writeln(I, ' ', NombreEnfermedadPt^[I]);}
            END;
{   writeln('Numero de enfermedades=', NumeroEnfermedades);}
        close(ff);
        reset(ff, Filename1);
        readln(ff, basepopulation);
        Basepopulation := Factorpopulation * Basepopulation;
{writeln('Basepopulation=', Basepopulation);}
        TheWindow := GetNewWindow(Escriturawindow, @WRecord, POINTER(-1));
        SetPort(theWindow);
        TextFace([bold]);
        TextMode(0);
        FOR I := 1 TO NumeroEnfermedades DO
```

```
            BEGIN
               ProbSintoma[I] := ProbSintomaTypeHdl(NewHandle(sizeof(ProbsintomaType)));
               getIndString(str1, TxtStrID, 20);
               Str := stringof(Str1, I : 4);
               TextBox(pointer(ord(@Str) + 1), length(Str), rectangle1, teJustCenter);
               Str := stringof(NombreEnfermedadPt^[I]);
               TextBox(pointer(ord(@Str) + 1), length(Str), rectangle2, teJustCenter);
               readln(ff, NombreEnfermedadPt^[I]);
{writeln(NombreEnfermedadPt^[I]);}
               read(ff, Pro);
               FOR k1 := 1 TO NumSintomasEnfermedadPt^[I] DO
                  BEGIN
                     read(ff, Sintoma[K1]);
                     read(ff, Pr[K1]);
{Write(Sintoma[K1] : 3, Pr[K1] : 6 : 2);}
                  END;
               readln(ff, Sintoma[1]);
{writeln(' ');}
               Ocurrencias := TRUNC(exp((NumSintomasEnfermedadPt^[I]) * Ln(2.0)) + 0.1);
{   writeln( ' Enfermedad ', I : 2, ' : ', Ocurrencias : 4, ' Ocurrencias'); }
               OcurrenciasEnfermedad[I] := Ocurrencias;
               TotalSize := Ocurrencias * SizeOf(Probtype);
               Frecuencia[I] := FrecuenciaTypeHdl(NewHandle(TotalSize));
               Hlock(Handle(Frecuencia[I]));
               FrecuenciaPtr := Frecuencia[I]^;
               FOR L := 0 TO Ocurrencias - 1 DO
                  BEGIN
{bitsdesc(L, NumSintomasEnfermedad[I]);}
                     S := 1.0;
                     FOR L1 := 1 TO NumSintomasEnfermedadPt^[I] DO
                        BEGIN
                           IF BitTst(@L, 32 - L1) THEN
                              BEGIN
                                 S := S * Pr[L1];
                              END
                           ELSE
                              S := S * (1.0 - Pr[L1]);
                        END;
                     FrecuenciaPtr^[L + 1] := Trunc(S * Pro / S1 * Basepopulation + 0.5);
{writeln('I,L+1 =', I, L + 1, TRUNC(FrecuenciaPtr^[L + 1]));}
                  END;
               HUnlock(Handle(Frecuencia[I]));
{escribir;}
            END;
         FOR I := 1 TO NumeroEnfermedades DO
            BEGIN
               Hlock(handle(probSintoma[I]));
               FOR j := 1 TO NumSintomas DO
                  BEGIN
                     ProbSintoma[I]^^[J] := FIJ[J];
                  END;
               Hunlock(handle(probSintoma[I]));
            END;
{Escritura;}
         DisposeWindow(TheWindow);
         close(ff);
         Handlesdisposed := False;
1 :
         SetCursor(Arrow);
{datos_experto;}
      END;
END.
```

Unit ASIGNACIÓN

```
UNIT asignacion;

INTERFACE
   USES
      SkelGlobals, Inicializar, experto;

   PROCEDURE ModificarFrecuencia (I9 : longint; F1, F2 : extended);
   FUNCTION FrecuenciaSuceso (I9 : longint) : extended;
   PROCEDURE InicializarFrecuencias;
   PROCEDURE EscribirFrecuencia;

IMPLEMENTATION
   CONST
      numeromaxSintomas = 10;
   TYPE
      frecType = ARRAY[1..1024] OF extended;
      radioButtonsType = RECORD
            theControls : ARRAY[1..3] OF controlHandle;
            Active : integer;
         END;
      radiotype = ARRAY[1..numeromaxSintomas] OF radioButtonstype;
   VAR
      NOcurrencias : integer;
      Frecu : frecType;
      I9 : longint;
      ch : char;
      ch1 : STRING;
      P7 : real;
      r : Rect;
      Esta : boolean;
      Base, fact, Baseant : extended;
      fact1 : longint;

      radioButtons : radiotype;
      done, controlsinitialized : boolean;
      theWindow, whatWindow : windowPtr;
      queControl : controlHandle;
      windowRect : rect;
      theEvent : eventRecord;
      NUMEROSINTOMAS : INTEGER;

   FUNCTION NumintegerStr (Str : str255; Lengthmax : integer) : str255;
      VAR
         I, L, K, J : integer;
         Str1 : str255;
   BEGIN
      L := length(Str);
      IF L > 0 THEN
         BEGIN
            IF L > lengthmax THEN
               BEGIN
                  str := copy(str, 1, lengthmax);
                  L := Lengthmax;
               END;
            IF L > 0 THEN
               BEGIN
                  done := false;
```

```
                J := 0;
                str1 :='
                I := 1;
                WHILE (I <= L) DO
                   BEGIN
                      K := ord(str[I]);
                      IF ((K > 47) AND (K < 58)) THEN
                         BEGIN
                            J := J + 1;
                            str1[J] := str[I];
                         END;
                      I := I + 1;
                   END
             END;
          NumintegerStr := Copy(Str1, 1, J);
       END
    ELSE
       NumintegerStr := str;
END;

FUNCTION NumrealStr (Str : str255; lengthmax : integer) : str255;
   VAR
      I, L, K, J : integer;
      Str1 : str255;
      punto : boolean;
BEGIN
   L := length(Str);
   IF L > 0 THEN
      BEGIN
         IF L > lengthmax THEN
            BEGIN
               str := copy(str, 1, lengthmax);
               L := Lengthmax;
            END;
         punto := false;
         IF L > 0 THEN
            BEGIN
               done := false;
               J := 0;
               str1 :='
               I := 1;
               WHILE (I <= L) DO
                  BEGIN
                     K := ord(str[I]);
                     IF ((K = 46) OR ((K > 47) AND (K < 58))) AND NOT ((K = 46) AND (I = 1)) THEN
                        BEGIN
                           IF (k <> 46) OR NOT punto THEN
                              BEGIN
                                 J := J + 1;
                                 str1[J] := str[I];
                              END;
                           IF k = 46 THEN
                              punto := true;
                        END;
                     I := I + 1;
                  END
            END;
         NumrealStr := Copy(Str1, 1, J);
      END
   ELSE
      NumrealStr := Str;
END;
```

```
PROCEDURE GenerarIndependencia;
   VAR
      I9 : longint;
      J : integer;
      D : extended;
BEGIN
   FOR I9 := 0 TO NOcurrencias - 1 DO
      BEGIN
         D := Basepopulation;
         FOR J := 1 TO NUMEROSINTOMAS DO
            IF BitTst(@I9, 32 - J) THEN
               D := D * 0.8
            ELSE
               D := D * 0.2;
         Frecu[I9 + 1] := trunc(D);
      END;
END;

PROCEDURE EnableDItem (s : DialogPtr; Item : integer; estado : boolean);
   VAR
      stype : integer;
      shdl : Handle;
      SRect : Rect;
BEGIN
   getDItem(s, Item, stype, sHdl, sRect);
   Hlock(Handle(shdl));
   IF estado THEN
      HiliteControl(controlhandle(sHdl), 0)
   ELSE
      HiliteControl(controlhandle(sHdl), 255);
   Hunlock(Handle(shdl));
END;

PROCEDURE EscribirFrecuencia;
   VAR
      J : longint;
      S : extended;
BEGIN
   s := 0.0;
   FOR J := 1 TO NOcurrencias DO
      BEGIN
         write(trunc(Frecu[J] / Fact) : 8);
         s := S + Frecu[J];
      END;
   writeln(' ', trunc(S) : 8);
END;

FUNCTION FrecuenciaSuceso;
   VAR
      P5 : extended;
      J : integer;
      J9 : longint;
      K9 : longint;
BEGIN
   K9 := Hiword(I9);
   I9 := LoWord(I9);
   P5 := 0;
   FOR J9 := 0 TO NOcurrencias - 1 DO
      IF (BitOr(K9, J9) = BitOr(K9, I9)) THEN
         P5 := P5 + Frecu[J9 + 1];
   FrecuenciaSuceso := P5;
```

```
    END;

    PROCEDURE ModificarFrecuencia;
       VAR
          J9, K9 : longint;
    BEGIN
       K9 := Hiword(I9);
       I9 := LoWord(I9);
{writeln('F1,F2=', F1 : 15 : 10, F2 : 15 : 10);}
       FOR J9 := 0 TO NOcurrencias - 1 DO
          IF (BitOr(K9, J9) = BitOr(K9, I9)) THEN
             Frecu[J9 + 1] := Frecu[J9 + 1] * F1
          ELSE
             Frecu[J9 + 1] := Frecu[J9 + 1] * F2;
    END;

    PROCEDURE draw_window (Numerosintomas, Enfermedad : integer);
       VAR
          CancelRect, rectSintoma, rectTitle, r : rect;
          j, i : integer;
          str : STRING[60];
          Str1, Str2 : Str255;
    BEGIN
       SetPort(theWindow);
       IF NOT controlsinitialized THEN
          KillControls(theWindow);
       SetRect(recttitle, 10, 5, 390, 35);
       TextFace([bold]);
       getIndString(Str2, TxtStrID, 54);
       Str := Concat(Str2, NombreEnfermedadPt^[Enfermedad]);
       TextBox(pointer(ord(@Str) + 1), length(Str), Recttitle, teJustCenter);

       MoveTo(15, 40);
       getIndString(Str2, TxtStrID, 55);
       DrawString(Str2);
       MoveTo(378, 40);
       getIndString(Str2, TxtStrID, 56);
       DrawString(Str2);
       TextFace([]);
       TextSize(10);
       SetRect(rectSintoma, 10, 49, 370, 63);
       FOR J := 1 TO numeroSintomas DO
          BEGIN
             Str1 := NombresintomasPt^[Sintomasenfermedad[Enfermedad]^^[J]];
             Str := Copy(Str1, 1, 60);
             IF length(Str1) > 60 THEN
                BEGIN
                   Str[60] := '.';
                   Str[59] := '.';
                   Str[58] := '.';
                END;
             TextBox(pointer(ord(@Str) + 1), length(Str), Rectsintoma, teJustLeft);
             IF NOT controlsinitialized THEN
                BEGIN
                   WITH rectSintoma DO
                      SetRect(r, 378, top, 392, bottom);
                   FOR i := 1 TO 3 DO
                      BEGIN
                         IF i = RadioButtons[j].active THEN
                            RadioButtons[j].theControls[i] := NewControl(theWindow, r, '', true, 1, 0,
1, 2, j * 100 + i)
                         ELSE
```

```
                        RadioButtons[j].theControls[i] := NewControl(theWindow, r, '', true, 0, 0,
1, 2, j * 100 + i);
                    offSetRect(r, 25, 0);
                END;
            END;
        offSetRect(rectSintoma, 0, 18);
    END;
  IF controlsinitialized THEN
    DrawControls(theWindow);
  controlsinitialized := true;
  TextSize(12);
END;

PROCEDURE trata_radioButton (control : controlHandle);
  VAR
    queSintoma, queRespuesta : integer;
    refCon : longInt;
BEGIN
  refCon := GetCRefCon(control);
  queSintoma := refCon DIV 100;
  queRespuesta := refCon - queSintoma * 100;
  WITH RadioButtons[queSintoma] DO
    BEGIN
      SetCtlValue(theControls[Active], 0);
      SetCtlValue(theControls[queRespuesta], 1);
      Active := queRespuesta;
    END;
END;

PROCEDURE InicializarFrecuencias;
  CONST
    dialogId = 264;
    AlertFrecuenciayaDefinidaId = 265;
    sOK = 1;
    sFrecuencia = 2;
    seditableFrecuencia = 3;
    sProbabilidad = 4;
    sEditableProbabilidad = 5;
    sPoblacion = 6;
    sStaticpoblacion = 7;
    sPrevious = 8;
    sNext = 9;
    sCancel = 10;
    sNumeroenfermedad = 11;
    lengthmaxprob = 8;
    lengthmaxfrec = 8;
    lengthmaxnume = 4;
  VAR
    I, J, k : integer;
    Sdialog : DialogPtr;
    sitem, stype : integer;
    shdl : Handle;
    SRect : Rect;
    Str : str255;
    controlnumber, Alertoption : integer;
    Frec, Frecant : extended;
    Prob, Probant : extended;
    F1, F2 : extended;
    frecuenciaoption : boolean;
    l90 : longint;
    Enfermedad, NnEnfermedad : integer;
    Basepopulationant : extended;
```

```
   Postbase, prebase : longint;
   cambioenfermedad : boolean;

PROCEDURE AsignaUnos (I9 : longint);
   VAR
      J9, K9 : longint;
BEGIN
   K9 := Hiword(I9);
   I9 := LoWord(I9);
   Frecant := 0.0;
   FOR J9 := 0 TO NOcurrencias - 1 DO
      IF (BitOr(K9, J9) = BitOr(K9, I9)) THEN
         BEGIN
            Frecant := Frecant + 1;
            Frecu[J9 + 1] := 1.0;
         END;
   IF Base > 0.0 THEN
      Probant := Frecant / Base
   ELSE
      Probant := 0.0;
END;

PROCEDURE AsignaCompUnos (I9 : longint);
   VAR
      J9, K9 : longint;
BEGIN
   K9 := Hiword(I9);
   I9 := LoWord(I9);
   FOR J9 := 0 TO NOcurrencias - 1 DO
      IF (BitOr(K9, J9) <> BitOr(K9, I9)) THEN
         BEGIN
            Frecu[J9 + 1] := 1.0;
         END;
END;

PROCEDURE ActualizarDialogFrecuencia;
   VAR
      str1 : str255;
BEGIN
   str1 := stringof(Prob : 9 : 6);
   trim(str1);
   IF length(str1) = 0 THEN
      str1 := ' ';
   GetDItem(sdialog, sEditableProbabilidad, stype, shdl, srect);
   SetItext(shdl, Str1);
   str1 := Stringof(Trunc(Frec / Fact + 0.5));
   trim(str1);
   IF length(str1) = 0 THEN
      str1 := ' ';
   GetDItem(sdialog, seditableFrecuencia, stype, shdl, srect);
   SetItext(shdl, str1);
   str1 := Stringof(trunc(Base / Fact + 0.5));
   trim(str1);
   IF length(str1) = 0 THEN
      str1 := ' ';
   GetDItem(sdialog, sStaticpoblacion, stype, shdl, srect);
   SetItext(shdl, str1);
   str1 := Stringof(Enfermedad : 4);
   trim(str1);
   IF length(str1) = 0 THEN
      str1 := ' ';
   GetDItem(sdialog, snumeroenfermedad, stype, shdl, srect);
```

```
        SetItext(shdl, str1);
    END;

    PROCEDURE ActualizarFrecuencia;
        VAR
            I : integer;
    BEGIN
        I9 := 0;
        FOR I := 1 TO NUMEROSINTOMAS DO
            BEGIN
                IF RadioButtons[I].active = 1 THEN
                    BitSet(@I9, 32 - I);
                IF RadioButtons[I].active = 3 THEN
                    BitSet(@I9, 16 - I);
            END;
{writeln('I9=', I9);}
    END;

    PROCEDURE ActualizarValores (Enfermedad : integer);
        VAR
            J : integer;
    BEGIN
        FOR J := 1 TO NOcurrencias DO
            Frecuencia[Enfermedad]^^[J] := Trunc(Frecu[J]);
    END;

    PROCEDURE InicializarValores (Enfermedad : integer);
        VAR
            J, I : integer;
            r : rect;
    BEGIN
        Numerosintomas := NumSintomasEnfermedadPt^[Enfermedad];
        NOcurrencias := OcurrenciasEnfermedad[Enfermedad];
        controlsinitialized := false;
        IF NOcurrencias = 1 THEN
            BEGIN
                Generarindependencia;
            END
        ELSE
            BEGIN
                FOR J := 1 TO NOcurrencias DO
                    Frecu[J] := Frecuencia[Enfermedad]^^[J];
            END;
        FOR I := 1 TO NumeroSintomas DO
            RadioButtons[I].active := 3;
        frecuenciaoption := true;
        ActualizarFrecuencia;
        I90 := I9;
        Frecant := FrecuenciaSuceso(I9);
        Base := FrecuenciaSuceso(I90);
        Baseant := Base;
        IF Base <> 0.0 THEN
            Prob := Frecant / Base
        ELSE
            Prob := 0.0;
        Probant := Prob;
        Frec := Frecant;
        SetRect(r, 10, 45, 400, 243);
        EraseRect(r);
    END;

BEGIN
```

```
{EscribirFrecuencia;}
        str := ' ';
        Fact := Factorpopulation;
        setRect(WindowRect, 10, 30, 480, 330);
        theWindow := NewWindow(NIL, windowRect, '', true, 1, WindowPtr(-1), false, 0);
        SetPort(theWindow);
        Enfermedad := 1;
        InicializarValores(Enfermedad);
        Draw_Window(numeroSintomas, Enfermedad);
        Sdialog := GetNewDialog(DialogID, NIL, Pointer(-1));
        setdialogtext(sdialog, sfrecuencia, 71);
        setdialogtext(sdialog, sprobabilidad, 67);
        setdialogtext(sdialog, sPoblacion, 60);
        setdialogtext(sdialog, sprevious, 61);
        setdialogtext(sdialog, sNext, 62);
        setdialogtext(sdialog, scancel, 30);

        InitCursor;
        FlushEvents(everyEvent, 0);
        validRect(theWindow^.portRect);
        EnableDItem(Sdialog, Sprevious, false);
        done := false;
        ActualizarDialogFrecuencia;
        cambioenfermedad := false;
        REPEAT
           SystemTask;
           IF GetNextEvent(mDownMask + UpdateMask + keyDownMask, theEvent) THEN
              BEGIN
                 IF NOT IsDialogEvent(theEvent) THEN
                    CASE theEvent.what OF
                       MouseDown :
                          BEGIN
                             globalToLocal(theEvent.where);
                             ControlNumber := FindControl(theEvent.where, theWindow,
queControl);
{writeln('controlnumber=', controlnumber, ' rfcon=', queControl^^.contrlRfCon, ' (queControl =
OKButton)=', (queControl = OKButton));}
                             IF ControlNumber <> 0 THEN
                                BEGIN
                                   IF trackControl(queControl, theEvent.where, NIL) <> 0 THEN
                                      trata_radioButton(queControl);
                                   ActualizarFrecuencia;
                                   Frecant := FrecuenciaSuceso(I9);
                                   IF Base <> 0 THEN
                                      Probant := Frecant / Base
                                   ELSE
                                      Probant := 0.0;
                                   Frec := Frecant;
                                   Prob := Probant;
                                   ActualizarDialogFrecuencia;
                                END;
                          END;

                       UpdateEvt :
                          IF windowPtr(theEvent.message) = theWindow THEN
                             BEGIN
                                beginUpdate(theWindow);
                                Draw_window(numeroSintomas, Enfermedad);
                                EndUpdate(theWindow);
                             END;
                       keyDown :
                          BEGIN
```

```
            i := BitAnd(theEvent.message, charCodeMask);
            IF (i = 3) OR (i = 13) THEN
               done := true;
         END;
   END
ELSE IF DialogSelect(theEvent, sDialog, sItem) THEN
   BEGIN
      CASE SItem OF
         sNumeroenfermedad :
            BEGIN
               cambioenfermedad := true;
               getDitem(sdialog, snumeroenfermedad, stype, shdl, srect);
               Hlock(Handle(shdl));
               getItext(shdl, str);
               IF length(str) = 0 THEN
                  str := '';
               Str := NumintegerStr(Str, lengthmaxnume);
               trim(str);
               IF length(str) = 0 THEN
                  str := '';
               SetIText(shdl, Str);
               Hunlock(Handle(shdl));
            END;
         seditableFrecuencia :
            BEGIN
               GetDitem(Sdialog, seditableFrecuencia, stype, shdl, sRect);
               Hlock(Handle(shdl));
               GetIText(shdl, Str);
               IF (length(str) = 0) THEN
                  str := '';
               Str := NumintegerStr(Str, lengthmaxfrec);
               SetIText(shdl, Str);
               IF length(Str) = 0 THEN
                  str := '0';
               readString(str, frec);
               IF Frec < 0 THEN
                  SetItext(shdl, '');
               Frecuenciaoption := true;
               Hunlock(Handle(shdl));
            END;
         seditableProbabilidad :
            BEGIN
               GetDitem(Sdialog, seditableProbabilidad, stype, shdl, sRect);
               Hlock(Handle(shdl));
               GetIText(shdl, Str2);
               IF length(str2) = 0 THEN
                  str2 := '';
               Str2 := NumrealStr(Str2, lengthmaxprob);
               SetIText(shdl, Str2);
               IF Str2 = '' THEN
                  Str2 := '0.00';
               ReadString(Str2, Prob);
               IF (Prob < 0.0) THEN
                  SetItext(shdl, '0.0');
               IF Prob > 1.00 THEN
                  SetItext(shdl, '1.0');
               Frecuenciaoption := false;
               Hunlock(Handle(shdl));
            END;
         sOK :
            IF cambioenfermedad THEN
               BEGIN
```

```
        getDitem(sdialog, snumeroenfermedad, stype, shdl, srect);
        Hlock(Handle(shdl));
        getItext(shdl, str);
        IF length(str) > 0 THEN
            readstring(str, Nnenfermedad)
        ELSE
            Nnenfermedad := 1;
        IF Nnenfermedad < 1 THEN
            BEGIN
                Nnenfermedad := 1;
            END;
        IF NnEnfermedad > Numeroenfermedades THEN
            BEGIN
                NnEnfermedad := NumeroEnfermedades;
            END;
        Str := Stringof(Nnenfermedad : 4);
        trim(str);
        IF length(str) = 0 THEN
            str := '';
        SetIText(shdl, Str);
        Hunlock(Handle(shdl));
        IF NnEnfermedad <> Enfermedad THEN
            BEGIN
                ActualizarValores(enfermedad);
                Enfermedad := Nnenfermedad;
                InicializarValores(Enfermedad);
                Draw_window(numeroSintomas, Enfermedad);
                ActualizarDialogFrecuencia;
                IF Enfermedad <= 1 THEN
                    EnableDItem(Sdialog, sPrevious, false)
                ELSE
                    EnableDItem(Sdialog, sPrevious, true);
                IF Enfermedad >= Numeroenfermedades THEN
                    EnableDItem(Sdialog, sNext, false)
                ELSE
                    EnableDItem(Sdialog, sNext, true);
            END;
        cambioenfermedad := false;
    END
ELSE
    BEGIN
        IF frecuenciaoption THEN
            BEGIN
                GetDitem(Sdialog, seditableFrecuencia, stype, shdl, sRect);
                GetIText(shdl, Str);
                IF Str = '' THEN
                    Str := '0.00';
                ReadString(Str, Frec);
                Frec := Frec * Fact;
                Basepopulationant := Base;
                Prebase := trunc(Base + 0.5 * Factorpopulation);
                IF Frecant = 0 THEN
                    AsignaUnos(I9);
                F1 := Frec / Frecant;
                F2 := 1.0;
                ModificarFrecuencia(I9, F1, F2);
                Base := FrecuenciaSuceso(I90);
                PostBase := Trunc(base + 0.5 * Factorpopulation);
                IF Base > 0.0 THEN
                    Prob := Frec / Base
                ELSE
                    Prob := 0.0;
```

```
                    ActualizarDialogFrecuencia;
                    Frecant := Frec;
                    Probant := Prob;
                    Basepopulation := Basepopulation + (PostBase - Prebase);
{EscribirFrecuencia;}
                END
            ELSE IF Base > 0 THEN
                BEGIN
                    GetDitem(Sdialog, seditableProbabilidad, stype, shdl,
sRect);
                    Hlock(Handle(shdl));
                    GetIText(shdl, Str2);
                    IF Str2 = '' THEN
                        Str2 := '0.00';
                    ReadString(Str2, Prob);
                    IF (I9 = I90) AND (Prob <> 1.0) THEN
                        BEGIN
                            getIndString(Str2, TxtStrID, 58);
                            Alerta(Str2);
                            SetItext(shdl, '1.00');
                            Prob := 1.00;
                            Probant := Prob;
                        END
                    ELSE
                        BEGIN
                            IF (I9 <> I90) OR (Prob <> 1.0) THEN
                                BEGIN
                                    IF Probant <> 1.0 THEN
                                        BEGIN
                                            IF Probant = 0.0 THEN
                                                AsignaUnos(I9);
                                            F1 := Prob / probant;
                                            F2 := (1.0 - Prob) / (1.0 - Probant);
                                            ModificarFrecuencia(I9, F1, F2);
                                            Frec := FrecuenciaSuceso(I9);
                                            Base := FrecuenciaSuceso(I90);
                                            ActualizarDialogFrecuencia;
                                            Frecant := Frec;
                                            Probant := Prob;
{EscribirFrecuencia;}
                                        END
                                    ELSE
                                        BEGIN
                                            AsignaCompUnos(I9);
                                            Frec := FrecuenciaSuceso(I9);
                                            Base := FrecuenciaSuceso(I90);
                                            Probant := Frec / Base;
                                            F1 := Prob / probant;
                                            F2 := (1.0 - Prob) / (1.0 - Probant);
                                            ModificarFrecuencia(I9, F1, F2);
                                            Frec := FrecuenciaSuceso(I9);
                                            Base := FrecuenciaSuceso(I90);
                                            ActualizarDialogFrecuencia;
                                            Frecant := Frec;
                                            Probant := Prob;
                                        END;
                                END;
                        END;
                    Hunlock(Handle(shdl));
                END;

        END;
```

```
                        sPrevious :
                            BEGIN
                                IF Enfermedad <= 2 THEN
                                    BEGIN
                                        EnableDItem(Sdialog, Sprevious, false);
                                    END;
                                ActualizarValores(enfermedad);
                                Enfermedad := Enfermedad - 1;
                                InicializarValores(Enfermedad);
                                Draw_window(numeroSintomas, Enfermedad);
                                EnableDItem(Sdialog, SNext, True);
                                ActualizarDialogFrecuencia;
                            END;
                        SNext :
                            BEGIN
                                IF Enfermedad >= Numeroenfermedades - 1 THEN
                                    BEGIN
                                        EnableDItem(Sdialog, sNext, false);
                                    END;
                                ActualizarValores(Enfermedad);
                                Enfermedad := Enfermedad + 1;
                                InicializarValores(Enfermedad);
                                Draw_window(numeroSintomas, Enfermedad);
                                EnableDItem(Sdialog, sPrevious, true);
                                ActualizarDialogFrecuencia;
                            END;
                        sCancel :
                            BEGIN
                                done := true;
                            END;
                        OTHERWISE
                    END;
                END;
            END;
        UNTIL done;
        disposdialog(sdialog);
        KillControls(theWindow);
        closeWindow(theWindow);
        ActualizarValores(Enfermedad);
    END;
END.
```

Unidad SKELMSGPROCS

```
UNIT SkelMsgProcs;

INTERFACE
   USES
      MacExpress, SkelGlobals;

   PROCEDURE StdMXMessageProc (theWindow : MXPeek; VAR theEvent : EventRecord);
   PROCEDURE StdPlMessageProc (thePanel : PanelHandle;VAR theEvent : EventRecord);
   PROCEDURE StdVwMessageProc (theView : ViewHandle; VAR theEvent : EventRecord);

IMPLEMENTATION

   PROCEDURE MyMXGoAway (theWindow : WindowPtr);
   BEGIN
      DisposeIcon(GetWindsIcon(theWindow));
      MXDisposeWindow(theWindow);
   END; { procedure MyMXGoAway }

   PROCEDURE StdMXMessageProc;
   BEGIN
      CASE theEvent.what OF
         msgMDown, msgMUp :
            StdMXMouseMsg(theWindow, theEvent);
         msgKeyDown, msgKeyUp, msgAutoKey :
            StdMXKeyMsg(theWindow, theEvent);
         msgUpdate :
            StdMXUpdateMsg(theWindow, theEvent);
         msgDrag :
            StdMXDragMsg(theWindow, theEvent);
         msgGrow :
            StdMXGrowMsg(theWindow, theEvent);
         msgGoAway :
            MyMXGoAway(POINTER(theWindow));
         msgCursor :
            StdMXCursorMsg(theWindow, theEvent);
         OTHERWISE
            StdMXOtherMsg(theWindow, theEvent);
      END;
   END; { procedure StdMXMessageProc }

   PROCEDURE StdPlMessageProc;
   BEGIN
      CASE theEvent.what OF
         msgDrawStruc :
            StdPIDStrucMsg(thePanel, theEvent);
         msgDrawCont :
            StdPIDContMsg(thePanel, theEvent);
         msgHitCont :
            StdPlMouseMsg(thePanel, theEvent);
         OTHERWISE
            StdPIOtherMsg(thePanel, theEvent);
      END;
   END; { procedure StdPlMessageProc }

   PROCEDURE StdVwMessageProc;
      VAR
         newWhere : Point;
```

```
BEGIN
   CASE theEvent.what OF
      msgNull :
         BEGIN
   { Do idle activities }
         END;
      msgKeyDown, msgKeyUp, msgAutoKey :
         BEGIN
   { Respond to keystrokes }
         END;
      msgActivate :
         BEGIN
            StdVwActivateMsg(theView, theEvent);
   { Highlight your selection(s) }
         END;
      msgDrawStruc :
         StdVwDStrucMsg(theView, theEvent);
      msgDrawCont :
         BEGIN
            StdVwDContMsg(theView, theEvent);
   { Draw your pad }
         END;
      msgHitStruc :
         StdVwHStrucMsg(theView, theEvent);
      msgHitCont :
         BEGIN
            AutoScroll(theView, TRUE);
            WHILE StillDown DO
               BEGIN
                  GetMouse(newWhere);
      { Respond to mouse location }
               END;
            AutoScroll(theView, FALSE);
         END;
      msgUpScroll, msgDownScroll, msgUpPage, msgDownPage, msgThumb, msgScrollTo :
         StdVwScrollMsg(theView, theEvent);
      msgCursor :
         BEGIN
            IF PtInRgn(theEvent.where, theView^^.vwContRgn) THEN
               BEGIN
                  LocalToPad(theEvent.where);
                  IF PtInRect(theEvent.where, theView^^.vwPadRect) THEN
                     SetCursor(arrow)  { put special cursor here }
                  ELSE
                     SetCursor(arrow);
               END
            ELSE
               SetCursor(arrow);
         END;
      msgTask :
         BEGIN
   { Do your task }
         END;
      OTHERWISE
         ;   { Do nothing but prevent range check exceptions }
   END;
END; { procedure StdVwMessageProc }
END.
```

Unidad INITIAL

```
UNIT initial;

INTERFACE
   USES
      SkelGlobals;

   PROCEDURE Inicializa;

IMPLEMENTATION

   PROCEDURE Inicializa;
   BEGIN
      Inicializado := false;
      handlesdisposed := True;
      NumsintomasEnfermo := 0;
      Enfermosimulado := False;
      OptionProbabilidad := True;
      OptionVerosimilitud := False;
      OptionHelp := False;
      OptionProbSintomasNoRelevantes := False;
      OptionSimulation := False;
      Diagnosisallowed := False;
      Sintomasdialogdisposed := True;
      Numsintomas := 0;
      Numeroenfermedades := 0;
      Basepopulation := 0;
   END;
END.
```

Unidad SKELCOMMAND

```
UNIT SkelCommand;

INTERFACE

    USES
        MacExpress, SkelGlobals, experto, SkelMsgProcs, display_list, Inicializar, Asignacion, Initial,
inicializacion, talk;

    FUNCTION CommandNumber (theMenuID, theItem : INTEGER) : Command;
    FUNCTION CanDoCommand (cmdNumber : Command; VAR markChar : CHAR;
                           VAR itemText : Str255) : BOOLEAN;
    FUNCTION DoCommand (cmdNumber : Command) : BOOLEAN;
    PROCEDURE do_startup;

IMPLEMENTATION

    FUNCTION CommandNumber;
    BEGIN
        CommandNumber := 0;
        CASE theMenuID OF
            mApple :
                CASE theItem OF
                    1 :
                        CommandNumber := cAboutAppl;
                    OTHERWISE
                        CommandNumber := cDeskAcc;
                END;
            mFile :
                CASE theItem OF
                    1 :
                        CommandNumber := cOpen;
                    11 :
                        CommandNumber := cQuit;
                    OTHERWISE
                        CommandNumber := theMenuID + theItem;
                END;
            OTHERWISE
                CommandNumber := theMenuID + theItem;
        END;
    END; { function CommandNumber }

    FUNCTION CanDoCommand;
        VAR
            fWindow : WindowPtr;
            itemStr : StringHandle;
    BEGIN
        fWindow := FrontWindow;
        CanDoCommand := TRUE;   { Default to TRUE }
        IF NOT SysWindInFront THEN
            IF (cmdNumber >= cUndo) AND (cmdNumber <= cClear) THEN
                CanDoCommand := FALSE;
        IF OptionHelp = True THEN
            BEGIN
                candoCommand := True;
                IF cmdNumber = cHelp THEN
                    MarkChar := chr(checkMark);
            END
```

```
ELSE
  BEGIN
    CASE cmdNumber OF
      cAboutAppl :
        getIndString(itemText, TxtStrID, 24);
      cOpen :
        CanDoCommand := TRUE;
      cInicenfermedades :
        IF Numsintomas <= 0 THEN
          CanDoCommand := false;
      CInicFrecuencias :
        IF NumeroEnfermedades <= 0 THEN
          CanDoCommand := false;
      cDiagnosis, cActBase, cCriterioDiagnostico, cSimulacion, cSimularEnfermo :
        BEGIN
          IF NOT inicializado THEN
            CanDoCommand := false
        END;
      cProbabilidades :
        BEGIN
          IF NOT inicializado THEN
            CanDoCommand := false
          ELSE IF optionProbabilidad THEN
            MarkChar := chr(checkMark);
        END;
      cverosimilitudes :
        BEGIN
          IF NOT inicializado THEN
            CanDoCommand := false
          ELSE IF optionVerosimilitud THEN
            MarkChar := chr(checkMark);
        END;
      cSintomasNoRelevantes :
        BEGIN
          IF NOT inicializado THEN
            CanDoCommand := false
          ELSE IF optionProbSintomasNoRelevantes THEN
            MarkChar := chr(checkMark);
        END;
      cTodosSintomas :
        BEGIN
          IF NOT inicializado THEN
            CanDoCommand := false
          ELSE IF OptionSimulation THEN
            MarkChar := chr(checkMark);
        END;
      cDiagAuto :
        IF NOT Diagnosisallowed OR (Numsintomasenfermo = 0) THEN
          CanDoCommand := false;
      cExplicar :
        IF NOT EndDiagnostico THEN
          CanDoCommand := false;
      cMostrarEnfermedad :
        CanDoCommand := Enfermosimulado;
      cHelp, CInfo :
        CanDoCommand := True;
      cEnfer, Csinto :
        BEGIN
          IF NOT inicializado THEN
            CanDoCommand := false;
        END;
      cIncEnferm :
```

```
                BEGIN
                    IF NOT inicializado THEN
                        CanDoCommand := false;
                    IF NumSintomasEnfermo = 0 THEN
                        CanDoCommand := false;
                END;
            cEspanol :
                IF idioma = espanol THEN
                    MarkChar := chr(checkMark);
            cIngles :
                IF idioma = ingles THEN
                    MarkChar := chr(checkMark);
            cHabla :
                BEGIN
                    IF idioma = Ingles THEN
                        CanDoCOmmand := true
                    ELSE
                        CanDoCommand := false;
                    IF talkOn THEN
                        MarkChar := chr(checkMark);
                END;
            OTHERWISE
                ;   { Do nothing but prevent range check exceptions }
        END;
    END;
END; { function CanDoCommand }

FUNCTION DoCommand;
    VAR
        windPtr : WindowPtr;
        iconHdl : IconHandle;
        aPanel : PanelHandle;
        aView : ViewHandle;
        aRect : Rect;
        aPoint : Point;
        itemHit : INTEGER;
        tempStr : Str255;
        Lastbutton : integer;
        dialogpointer : DialogPtr;
        MenuHdl : MenuHandle;
        Ch : Char;
        Dum : integer;
        question : cursHandle;
        Str1, Str2, fname : str255;
        tempHdl : handle;
        vol : integer;
        diagnosticofailed, dialogopen : boolean;
BEGIN
    DoCommand := TRUE;
    IF (OptionHelp = True) AND (cmdNumber MOD 100 <> 0) AND (cmdNumber <> cQuit) THEN
        BEGIN
            getIndString(str1, HelpStrID, 15);
            Str2 := '';
            CASE cmdNumber OF
                cOpen :
                    getIndString(str1, HelpStrID, 16);
                cDiagAuto :
                    BEGIN
                        getIndString(str1, HelpStrID, 3);
                        getIndString(str2, HelpStrID, 4);
                    END;
                cSintEnfer :
```

```
            getIndString(str1, HelpStrID, 5);
        Cinicializar :
            getIndString(str1, HelpStrID, 6);
        Cexplicar :
            getIndString(str1, HelpStrID, 25);
        cEnfer :
            getIndString(str1, HelpStrID, 1);
        cSinto :
            getIndString(str1, HelpStrID, 2);
        cHelp :
            BEGIN
                getIndString(str1, TxtStrID, 5);
                DisposDialog(HelpDialog);
                OptionHelp := False;
                SetCursor(Arrow);
            END;
        cActBase :
            getIndString(str1, HelpStrID, 7);
        cIncEnferm :
            BEGIN
                getIndString(str1, HelpStrID, 8);
                getIndString(str2, HelpStrID, 9);
            END;
        cInicfromfile :
            getIndString(str1, HelpStrID, 10);
        CInicSintomas :
            getIndString(str1, HelpStrID, 21);
        CinicEnfermedades :
            getIndString(str1, HelpStrID, 22);
        CInicFrecuencias :
            getIndString(str1, HelpStrID, 23);
        CInicPopulation :
            getIndString(str1, HelpStrID, 24);
        cProbabilidades :
            getIndString(str1, HelpStrID, 11);
        cverosimilitudes :
            getIndString(str1, HelpStrID, 12);
        cSintomasNoRelevantes :
            getIndString(str1, HelpStrID, 20);
        cSimularEnfermo :
            getIndString(str1, HelpStrID, 13);
        cMostrarEnfermedad :
            getIndString(str1, HelpStrID, 14);
        cTodosSintomas :
            getIndString(str1, HelpStrID, 19);
        cingles :
            getIndString(str1, HelpStrID, 18);
        cespanol :
            getIndString(str1, HelpStrID, 17);
        OTHERWISE
    END;
    dialogPointer := GetNewDialog(1530, NIL, POINTER(-1));
    ParamText(Str1, Str2, '', '');
    IF idioma <> espanol THEN
        BEGIN
            DrawDialog(dialogPointer);
            Limpiar(str1);
            Say(str1);
            Limpiar(str2);
            Say(str2)
        END;
    REPEAT
```

```
                ModalDialog(NIL, itemHit);
            UNTIL itemHit = 1;
            DisposDialog(dialogPointer);
        END
    ELSE
        BEGIN
            CASE cmdNumber OF
              · cAboutAppl :
                BEGIN
                    dialogPointer := GetNewDialog(256, NIL, POINTER(-1));
                    Set_dialog_text(dialogPointer, 3, 31);
                    Set_dialog_text(dialogPointer, 4, 32);
                    Set_dialog_text(dialogPointer, 5, 33);
                    Set_dialog_text(dialogPointer, 6, 34);
                    DrawDialog(dialogPointer);
                    HCenterWindow(dialogPointer, TRUE);
                    REPEAT
                        ModalDialog(NIL, itemHit);
                    UNTIL itemHit = OK;
                    DisposDialog(dialogPointer);
                END;
              cOpen :
                BEGIN
                    IF get_file_name(vol, fname) THEN
                        BEGIN
                            Disposeallhdl;
                            Inicializa;
                            Read_data(vol, fname);
                            OrdenarEnfermedades;
                            SetUpMenus(true);
                            EndDiagnostico := false;
                        END;
                END;
              cInicfromfile :
                BEGIN
                    Disposeallhdl;
                    Inicializa;
                    inicializar;
                    EndDiagnostico := false;
                    OrdenarEnfermedades;
                END;
              CInicsintomas :
                BEGIN
                    InicializarDatosSintomas;
                END;
              CInicenfermedades :
                BEGIN
                    InicializarDatosEnfermedades;
                    OrdenarEnfermedades;
                END;
              CInicFrecuencias :
                InicializarFrecuencias;
              CInicPopulation :
                BEGIN
                    inicializarpoblacion;
                END;
              cDiagAuto :
                BEGIN
                    EndDiagnostico := True;
                    RequiredDiagnostico := TRUE;
                    Cancel := False;
{InicializarSintomas;}
```

```
                    dialogopen := false;
                    REPEAT
                       IF RequiredDiagnostico THEN
                          BEGIN
                             Diagnostico(diagnosticofailed);
                             IF NOT diagnosticofailed THEN
                                BEGIN
                                   orden(dialogpointer);
                                   dialogopen := true;
                                END
                             ELSE
                                Enddiagnostico := false;
                          END;
                       IF NOT diagnosticofailed THEN
                          IF Cancel = False THEN
                             BEGIN
                                Requireddiagnostico := TRUE;
                                PreguntaSintoma(lastbutton);
                                IF lastbutton = 0 THEN
                                   EndDiagnostico := True;
                             END;
                    UNTIL EndDiagnostico OR diagnosticofailed;
                    IF dialogopen THEN
                       DisposDialog(dialogpointer);
                 END;
              Cinicializar :
                 BEGIN
                    InicializarSintomas;
                    EndDiagnostico := false;
                 END;
              cExplicar :
                 Explicar;
              cSintEnfer :
                 BEGIN
{InicializarSintomas;}
                    EndDiagnostico := false;
                    Sintomas(NumSintomas);
                    IF NumSintomasEnfermo > 0 THEN
                       Diagnosisallowed := TRUE;
                 END;
              CEnfer :
                 Information(NumeroEnfermedades, 1);
              Csinto :
                 Information(NumSintomas, 2);
              cActBase :
                 BEGIN
                    actualizar;
                 END;
              cIncEnferm :
                 BEGIN
                    EndDiagnostico := false;
                    EnfermoaBase;
                 END;
              cProbabilidades :
                 BEGIN
                    OptionProbabilidad := True;
                    OptionVerosimilitud := False;
                 END;
              cverosimilitudes :
                 BEGIN
                    OptionVerosimilitud := True;
                    OptionProbabilidad := False;
```

```
                END;
            cSintomasNoRelevantes :
                BEGIN
                    OptionProbSintomasNoRelevantes := NOT
OptionProbSintomasNoRelevantes;
                END;
            cSimularEnfermo :
                BEGIN
                    InicializarSintomas;
                    SimularEnfermo;
                    Enfermosimulado := True;
                    Diagnosisallowed := TRUE;
                    EndDiagnostico := false;
                END;
            cMostrarEnfermedad :
                MostrarEnfermedad;
            cTodosSintomas :
                OptionSimulation := NOT OptionSimulation;
            cHelp :
                BEGIN
                    getIndString(str, TxtStrID, 4);
                    ParamText('', Str, '', '');
                    Dum := NoteAlert(NoteAlertID, NIL);
                    OptionHelp := True;
                    HelpDialog := GetNewDialog(HelpdialogId, NIL, POINTER(-1));
                    Set_dialog_text(helpDialog, 1, 26);
                    HCenterWindow(HelpDialog, true);
                    DrawDialog(HelpDialog);
                    question := GetCursor(QuestionCursor);
                    SetCursor(question^^);
                    SetUpMenus(true);
                END;
            cEspanol :
                IF Idioma = ingles THEN
                    BEGIN
                        Idioma := espanol;
                        setMenuBar(menuEspanol);
                        DrawMenuBar;
                        SetUpMenus(true);
                        HelpStrID := 400;
                        TxtStrID := 500;
                    END;
            cIngles :
                IF idioma = espanol THEN
                    BEGIN
                        idioma := ingles;
                        setMenuBar(menuIngles);
                        drawMenuBar;
                        SetUpMenus(true);
                        HelpStrID := 1400;
                        TxtStrID := 1500;
                    END;
            cHabla :
                talkOn := NOT talkOn;
            OTHERWISE
                DoCommand := FALSE;
        END;
    END;
END; { function DoCommand }

PROCEDURE do_startup;
    LABEL
```

```
        1;
    VAR
        i, Count, Message : integer;
        theFile : appFile;
        bool : boolean;
BEGIN
    CountAppFiles(Message, Count);
    IF count <> 0 THEN
        BEGIN
            FOR i := 1 TO count DO
                BEGIN
                    GetAppFiles(i, theFile);
                    IF theFile.fType = DocType THEN
                        BEGIN
                            read_data(theFile.vRefNum, theFile.fname);
                            ClrAppFiles(i);
                            GOTO 1;
                        END;
                END;
1 :
            IF message = AppPrint THEN
                BEGIN
                    quitMX;
                    exitToShell
                END
        END
END;
END.
```

Programa SKELETON

```
PROGRAM Skeleton;
{$I-}

   USES
      MacExpress, SkelGlobals, SkelMsgProcs, SkelCommand, Experto, Initial, talk;

BEGIN { MAIN }
   MoreMasters;
   MoreMasters;
   MoreMasters;
   MoreMasters;
   MoreMasters;
   MoreMasters;
   MoreMasters;
   MoreMasters;
   MoreMasters;
   MoreMasters;
   MoreMasters;
   MoreMasters;
   MoreMasters;
   Idioma := espanol;
   HelpStrID := 400;
   TxtStrID := 500;
   New(NumSintomasEnfermedadPt);
   New(NombreSintomasPt);
   New(NombreEnfermedadPt);
   New(FrecuenciaPtr);
   New(SintomaEnfermedadPtr);
   hideAll;
   InitMX(@mxVars.applDone, @thePort, NIL);
   menuIngles := GetNewMBar(257);
   menuEspanol := GetNewMBar(256);
   Inicializa;
   InitTalk;
   talkOn := true;
   InicializarSintomas;
  { do our own intialization }
   do_startup;
   LaunchMX(NIL);
   QuitMX;
   dispose(NumSintomasEnfermedadPt);
   dispose(NombreSintomasPt);
   dispose(NombreEnfermedadPt);
   closeTalk;
END. { of Skeleton }
```

7.2.-LISTADO DEL PROGRAMA PARA CAMBIO DE ORDENACIÓN DE PSEUDOBÁSICOS

```
PROGRAM ordenacion;

   CONST
      KMAX = 5;
      KLIMIT = 32;    {2**KMAX}

   TYPE
      vector = ARRAY[1..KLIMIT] OF Longint;

   VAR
      rec : rect;
      I, R : integer;
      Basico, PseudoBasico, Generado : longint;
      NumBasicos, NumPseudoBasicos, NumGenerados, Num : integer;
      NumeroBasicos : integer;
      Basicos : Vector;
      c : char;



   FUNCTION power (A, B : real) : real;
   BEGIN
      power := exp(B * Ln(A));
   END;


   FUNCTION Base (N, A : Longint) : Vector;
{Descompone el conjunto con número de orden A en base N}
      VAR
         I : integer;
         J : Vector;
   BEGIN
{write('       Base ', N : 3, ' de ', A : 4, '     ');}
      FOR I := 1 TO NumBasicos - 1 DO
         BEGIN
            J[I] := A MOD N;
            A := A DIV N;
         END;
      J[NumBasicos] := A;
      Base := J;
{for I := 1 to NumBasicos do}
{write(J[I] : 3, ' ');}
{writeln;}
   END;

   FUNCTION Basico_PseudoBasico (A : longint) : longint;
      VAR
         L : longint;
         S : real;
         I : integer;
   BEGIN
      L := 0;
      S := 1 / 3;
      FOR I := 1 TO R DO
         BEGIN
            S := S * 3;
```

```
            IF BitTst(@A, 32 - I) THEN
                L := L + trunc(S);
        END;
    Basico_PseudoBasico := L;
END;

FUNCTION PseudoBasico_Basico (A : longint;VAR Num : integer) : Vector;
    VAR
        I, I1 : integer;
        J, J1 : Vector;
BEGIN
    J := Base(3, A);
    Num := 1;
    J1[1] := 0;
    FOR I := 1 TO KMax DO
        IF J[I] = 2 THEN
            BEGIN
                FOR I1 := 1 TO Num DO
                    BEGIN
                        J1[I1 + Num] := J1[I1];
                        BitSet(@J1[I1], 32 - I);
                    END;
                Num := Num * 2;
            END
        ELSE
            BEGIN
                IF J[I] = 1 THEN
                    FOR I1 := 1 TO Num DO
                        BitSet(@J1[I1], 32 - I);
            END;
    PseudoBasico_Basico := J1;
END;


FUNCTION Basico_GeneradoBasicos (A : longint) : Longint;
    VAR
        B : longint;
BEGIN
    Basico_GeneradoBasicos := trunc(power(2.0, A));
END;


FUNCTION GeneradoBasicos_Basico (A : longint;VAR J : integer) : Vector;
    VAR
        G : Vector;
        I : integer;
BEGIN
    J := 0;
    FOR I := 1 TO NumBasicos DO
        BEGIN
            IF BitTst(@A, 32 - I) THEN
                BEGIN
                    J := J + 1;
                    G[J] := I - 1;
                END;
        END;
    GeneradoBasicos_Basico := G;
END;

PROCEDURE EscribirBasico (A : Longint; K : integer);
    VAR
        I : integer;
```

```
    BEGIN
       FOR I := 1 TO K DO
          IF BitTst(@A, 32 - I) THEN
             write(' 1 ')
          ELSE
             write(' 0 ');
       writeln;
    END;


    PROCEDURE Leer (VAR A : longint; ValorMaximo : longint);
       VAR
          corr : boolean;
    BEGIN
       corr := false;
       REPEAT
          readln(A);
          IF (A >= 0) AND (A <= ValorMaximo - 1) THEN
             corr := true
          ELSE
             writeln('Error');
       UNTIL corr;
    END;

    PROCEDURE Escribir1 (A : Longint; K : integer);
       VAR
          J : integer;
          B : vector;
    BEGIN
       B := Base(3, A);
       FOR J := 1 TO R DO
          write(B[J] : 3, ' ');
       writeln;
    END;


BEGIN
    SetRect(rec, 50, 50, 400, 250);
    SetTextRect(rec);
    showtext;
    R := 3;
    NumPseudoBasicos := trunc(power(3.0, R));
    NumBasicos := trunc(power(2.0, R));
    NumGenerados := trunc(power(2.0, NumBasicos));
    writeln('NumBasicos, NumPseudoBasicos= ', NumBasicos, NumPseudoBasicos);
    REPEAT
       writeln('********** ALGORITMO 1 **********');
       write(' Da orden 1 del conjunto basico: ');
       Leer(Basico, NumBasicos);
       EscribirBasico(Basico, R);
       PseudoBasico := Basico_PseudoBasico(Basico);
       write(' El orden 2 del conjunto basico es: ');
       writeln(PseudoBasico);
       Escribir1(PseudoBasico, R);
       writeln;
       writeln('********** ALGORITMO 2 **********');
       write(' Da orden 2 del conjunto PseudoBasico: ');
       Leer(PseudoBasico, NumPseudoBasicos);
       Escribir1(PseudoBasico, R);
       Basicos := PseudoBasico_Basico(PseudoBasico, NumeroBasicos);
       write(' El orden 1 del conjunto PseudoBasico es: ');
       FOR I := 1 TO NumeroBasicos DO
```

```
            write(Basicos[I] : 3, ' ');
        writeln;
        FOR I := 1 TO NumeroBasicos DO
            EscribirBasico(Basicos[I], R);
        writeln;
        writeln('********** ALGORITMO 3 **********');
        write(' Da orden 1 del conjunto Basico: ');
        Leer(Basico, NumBasicos);
        EscribirBasico(Basico, R);
        Generado := Basico_GeneradoBasicos(Basico);
        write(' El orden 3 del conjunto Basico es: ');
        writeln(Generado : 3);
        EscribirBasico(Generado, NumBasicos);
        writeln;
        writeln('********** ALGORITMO 4 **********');
        write(' Da orden 3 del conjunto Generado: ');
        Leer(Generado, NumGenerados);
        EscribirBasico(Generado, NumBasicos);
        Basicos := GeneradoBasicos_Basico(Generado, Num);
        write(' El orden 1 del conjunto Generado es: ');
        FOR I := 1 TO Num DO
            write(Basicos[I] : 3, ' ');
        writeln;
        FOR I := 1 TO Num DO
            EscribirBasico(Basicos[I], R);
        writeln;
        writeln('Dar A para abortar, return para continuar');
        read(c);
        writeln;
    UNTIL (c = 'a') OR (c = 'A');
END.
```

7.3.-CONTROL COHERENCIA

```
PROGRAM LinPar;

  CONST
    M1max = 33;
    M2max = 32;
    Nterminosmax = 30;
    NmaxRestricciones = 30;

  TYPE
    Independiente = ARRAY[1..Nterminosmax] OF real;
    xtype = ARRAY[1..M2max, 1..M1max] OF real;
    TindependienteType = ARRAY[1..M2max] OF Independiente;
    RestriccionesType = ARRAY[1..NmaxRestricciones] OF Independiente;
    tipo = (menorigual, menor);
    contestaciontype = (si, no, nosabe);

  VAR
    textr : Rect;
    KK, Numerosucesos : integer;
    automatico, iescritura, iescriturarestricciones, bien : boolean;
    Tindependiente : TindependienteType;
    M1, M2, Nterminos : integer;
    I9, J9, K9, J2 : longint;
    M, N, M3, M22, M23, Nbits, J3, J88 : integer;
    X : xtype;
    error : boolean;
    B : ARRAY[1..M2max, 1..M1max] OF real;
    C : ARRAY[1..M1max] OF real;
    AuxInd : ARRAY[1..M2max] OF independiente;
    NRestriccionesMenorIgual, NRestriccionesMenor : integer;
    NN : ARRAY[1..M2max] OF integer;
    RestriccionesMenorIgual, RestriccionesMenor : RestriccionesType;
    Nter : integer;
    leer, escribir : integer;
    casos : ARRAY[1..2] OF integer;
    a : integer;
    done, done1 : boolean;
    err : Oserr;
    JJJ : integer;
    contestacion : contestaciontype;
    soluciones : ARRAY[1..20, 1..2] OF independiente;
    Nsoluciones : integer;
    sucesoNumber : integer;

    Str : STRING;
    Filename : ARRAY[1..3] OF STR255;
    inicializado, handlesdisposed : boolean;
    firsttime : boolean;
    VRefNum, refnum, PathRefNum : ARRAY[1..3] OF integer;
    drvNum : integer;
    volName : stringPtr;
    NRestriccionesInicialMenorIgual, NRestriccionesInicialMenor : integer;


  FUNCTION NumeroBits1 (JJJ : longint) : integer;
    VAR
      I, Nbits : integer;
  BEGIN
```

```
    Nbits := 0;
    FOR I := 1 TO Numerosucesos DO
      IF BitTst(@JJJ, 32 - I) THEN
        Nbits := Nbits + 1;
    NumeroBits1 := Nbits;
  END;

  FUNCTION siguiente (JJJJ : longint) : longint;
    VAR
      I : integer;
      done : boolean;
      Nbits : integer;

  BEGIN
    Nbits := NumeroBits1(JJJJ);
    done := false;
    REPEAT
      WHILE (JJJJ < N - 1) AND NOT done DO
        BEGIN
          JJJJ := JJJJ + 1;
          IF NumeroBits1(JJJJ) = Nbits THEN
            done := true;
        END;
      IF NOT done THEN
        BEGIN
          JJJJ := 0;
          Nbits := Nbits + 1;
        END;
    UNTIL done OR (Nbits > Numerosucesos);
    IF done THEN
      siguiente := JJJJ
    ELSE
      siguiente := 0;
  END;


  PROCEDURE ManipularMatriz;

    VAR
      I, J : integer;
      pivot : real;

  BEGIN
    FOR J := 1 TO N DO
      X[SucesoNumber + 3, J] := C[J];
    Tindependiente[SucesoNumber + 3, SucesoNumber + 1] := 1.0;
    NRestriccionesInicialMenorIgual := NRestriccionesInicialMenorIgual + 1;
    RestriccionesMenorIgual[NRestriccionesInicialMenorIgual, 1] := -1.0;
    FOR J := 2 TO Nterminos DO
      RestriccionesMenorIgual[NRestriccionesInicialMenorIgual, J] := 0.0;
    RestriccionesMenorIgual[NRestriccionesInicialMenorIgual, SucesoNumber + 1] := 1.0;
    FOR J := 1 TO Nsoluciones DO
      BEGIN
        NRestriccionesInicialMenorIgual := NRestriccionesInicialMenorIgual + 1;
        RestriccionesMenorIgual[NRestriccionesInicialMenorIgual] := Soluciones[J, KK];
        IF KK = 1 THEN
          RestriccionesMenorIgual[NRestriccionesInicialMenorIgual, SucesoNumber + 1] := 1.0
        ELSE
          RestriccionesMenorIgual[NRestriccionesInicialMenorIgual, SucesoNumber + 1] := -
1.0;
      END;
```

```
END;


PROCEDURE manipularmatriz1;
   VAR
      I, J : integer;
      pivot : real;
BEGIN
   FOR I := 3 TO N - sucesoNumber + 4 DO
      BEGIN
         pivot := x[I, sucesonumber] / C[sucesonumber];
         FOR j := 1 TO sucesoNumber DO
            x[i, J] := x[i, j] - pivot * c[J];
         x[i, sucesonumber] := 0.0;
         Tindependiente[I, N - sucesoNumber + 2] := pivot;
      END;
   x[N - sucesoNumber + 4, sucesoNumber] := 1.0;
END;


PROCEDURE Errordisk (error : integer);

   VAR
      Dum : integer;

BEGIN
   IF (error <> noErr) AND (error <> opWrErr) THEN
      BEGIN
         Str := '';
         Str := Stringof(Str, error);
         ParamText('', Str, '', '');
         writeln('error=', Str);
         SysBeep(5);
         ExitToShell;
      END;
END;


PROCEDURE Writedata (fileNumber : integer);

   LABEL
      1;
   VAR
      pt : point;
      TypeList : SFTypeList;
      reply : SFReply;
      I, J : integer;
      reloj : cursHandle;
      Temp : Str255;
      Error : integer;
      Dum : integer;
      Er : OSErr;

   PROCEDURE WriteByte (what : ptr;
                        num : longInt);
      VAR
         error2 : integer;
   BEGIN
      error := FSWrite(refNum[fileNumber], num, what);
      IF error <> noErr THEN
         BEGIN
```

```
            error2 := FSClose(refNum[fileNumber]);
            ErrorDisk(error);
        END;
    END;

BEGIN

    reloj := GetCursor(WatchCursor);
    SetCursor(reloj^^);
    IF iescritura THEN
        BEGIN
            write('Graba datos en disco');
            writeln('M3=', M3);
            writeln(' Done=', Done);
            writeln('Done1=', Done1);
            writeln('NRestriccionesMenorIgual=', NRestriccionesMenorIgual);
            writeln('NRestriccionesMenor=', NRestriccionesMenor);
        END;
    WriteByte(@M3, sizeof(M3));
    WriteByte(@Done, sizeof(Done));
    WriteByte(@Done1, sizeof(Done1));
    WriteByte(@NRestriccionesMenorIgual, sizeof(NRestriccionesMenorIgual));
    WriteByte(@NRestriccionesMenor, sizeof(NRestriccionesMenor));
    WriteByte(@X, sizeOf(X));
    WriteByte(@Tindependiente, sizeOf(Tindependiente));
    WriteByte(@RestriccionesMenorIgual, sizeof(RestriccionesMenorIgual));
    WriteByte(@RestriccionesMenor, sizeof(RestriccionesMenor));

1 :
    SetCursor(Arrow);
END;

PROCEDURE readdata (fileNumber : integer);


    VAR
        I, J : integer;
        reloj : cursHandle;
        Temp : Str255;
        Error : OsErr;
        MyDialog : DialogPtr;
        TheItem : integer;
        Dum : integer;
        L : longint;


    PROCEDURE ReadByte (what : ptr;
                        num : longInt);
        VAR
            error2 : integer;
    BEGIN
        error := FSREad(refNum[fileNumber], num, what);
        IF error <> noErr THEN
            BEGIN
                error2 := FSClose(refNum[fileNumber]);
                ErrorDisk(error);
            END;
    END;

BEGIN
    ReadByte(@M3, sizeof(M3));
    IF iescritura THEN
```

```
        BEGIN
            write('Lee datos de disco');
            writeln('M3=', M3);
        END;
    ReadByte(@Done, sizeof(Done));
    IF iescritura THEN
        writeln(' Done=', Done);
    ReadByte(@Done1, sizeof(Done1));
    IF iescritura THEN
        writeln('Done1=', Done1);
    ReadByte(@NRestriccionesMenorIgual, sizeof(NRestriccionesMenorIgual));
    IF iescritura THEN
        writeln('NRestriccionesMenorIgual=', NRestriccionesMenorIgual);
    ReadByte(@NRestriccionesMenor, sizeof(NRestriccionesMenor));
    IF iescritura THEN
        writeln('NRestriccionesMenor=', NRestriccionesMenor);
    ReadByte(@X, sizeOf(X));
    ReadByte(@Tindependiente, sizeOf(Tindependiente));
    ReadByte(@RestriccionesMenorIgual, sizeof(RestriccionesMenorIgual));
    ReadByte(@RestriccionesMenor, sizeof(RestriccionesMenor));
END;

PROCEDURE RebobinarAll;
    VAR
        Er : OSErr;
BEGIN

    a := leer;
    leer := escribir;
    escribir := a;
    Er := SetFPos(RefNum[escribir], FsFromStart, 0);
    ErrorDisk(er);
    Er := SetFPos(RefNum[leer], FsFromStart, 0);
    ErrorDisk(er);
END;

FUNCTION StrInd (A : independiente) : STRING;
    VAR
        J : integer;
        st : STRING;
        blank : boolean;

BEGIN
    St := '';
    blank := true;
    IF (A[1] > 1.0e-05) OR (A[1] < -1.0e-05) THEN
        BEGIN
            St := concat(St, Stringof(A[1] : 4 : 1));
            blank := false;
        END
    ELSE
        St := concat(St, '    ');
    FOR J := 2 TO Nterminos DO
        BEGIN
            IF (A[J] > 1.0e-05) OR (A[J] < -1.0e-05) THEN
                BEGIN
                    IF a[J] > 0.0 THEN
                        St := Concat(St, '+')
                    ELSE
                        St := Concat(St, ' ');
                    blank := false;
                END
```

```
            ELSE
                St := concat(St, ' ');
            IF (A[J] > 1.0e-05) OR (A[J] < -1.0e-05) THEN
                BEGIN
                    St := Concat(St, Stringof(A[J] : 4 : 1), char(J + 63));
                    blank := false;
                END
            ELSE
                St := concat(St, '    ');
        END;
    IF blank THEN
        St := '0';
    StrInd := St;
END;


PROCEDURE WriteRestricciones (I, K : integer);
    VAR
        J : integer;
BEGIN
    FOR J := I TO K DO
        writeln(strInd(RestriccionesMenorIgual[J]), ' <= 0 ');
    FOR J := NRestriccionesInicialMenor + 1 TO NrestriccionesMenor DO
        writeln(strInd(RestriccionesMenor[J]), ' < 0 ');
END;

FUNCTION Igual (a, b : Independiente) : boolean;
    VAR
        J : integer;
        igual1 : boolean;
BEGIN
    Igual1 := true;
    J := 1;
    WHILE ((J < Nterminos) OR (J = Nterminos)) AND (Igual1 = true) DO
        BEGIN
            IF (A[J] - B[J] > 1.0e-6) OR (A[J] - B[J] < -1.0e-6) THEN
                Igual1 := false;
            J := J + 1;
        END;
    igual := igual1;
END;

PROCEDURE Normalizar (VAR A : independiente);

    VAR
        I : integer;
        max, au : real;

BEGIN
    Max := A[1];
    FOR I := 2 TO Nterminos DO
        BEGIN
            au := Abs(A[I]);
            IF Au > Max THEN
                Max := Au;
        END;
    FOR I := 1 TO Nterminos DO
        A[I] := A[I] / Max;
END;

PROCEDURE GuardarenLista (A : independiente;
                          t : tipo);
```

```
BEGIN
   CASE t OF
      Menorigual :
         BEGIN
            NrestriccionesMenorIgual := NrestriccionesMenorIgual + 1;
            RestriccionesMenorIgual[NrestriccionesMenorIgual] := A;
         END;
      menor :
         BEGIN
            NrestriccionesMenor := NrestriccionesMenor + 1;
            RestriccionesMenor[NrestriccionesMenor] := A;
         END;
   END;
END;

FUNCTION SeVerifica (A : independiente;
                     t : tipo) : contestaciontype;

   VAR
      I : integer;
      verifica : contestaciontype;

BEGIN
   verifica := Nosabe;
   I := 1;
   CASE t OF
      menorIgual :
         WHILE (I <= NrestriccionesMenorIgual) AND (verifica = nosabe) DO
            BEGIN
               IF Igual(RestriccionesMenorIgual[I], A) THEN
                  verifica := si;
               I := I + 1;
            END;
      menor :
         WHILE (I <= NrestriccionesMenor) AND (verifica = nosabe) DO
            BEGIN
               IF Igual(RestriccionesMenor[I], A) THEN
                  verifica := si;
               I := I + 1;
            END;
   END;
   IF (verifica = nosabe) THEN
      BEGIN
         FOR I := 1 TO Nterminos DO
            A[I] := -A[I];
         I := 1;
         CASE t OF
            Menor :
               WHILE (I <= NrestriccionesMenorIgual) AND (verifica = nosabe) DO
                  BEGIN
                     IF Igual(RestriccionesMenorIgual[I], A) THEN
                        verifica := no;
                     I := I + 1;
                  END;
            menorIgual :
               WHILE (I <= NrestriccionesMenor) AND (verifica = nosabe) DO
                  BEGIN
                     IF Igual(RestriccionesMenor[I], A) THEN
                        verifica := no;
                     I := I + 1;
                  END;
```

```
            END;
        END;
    Severifica := verifica;
END;

FUNCTION Minimo : integer;

    VAR
        I, J, K, mini, Mini0 : integer;
        aux, min, min0 : Independiente;
        Done : boolean;

BEGIN
    mini := NN[1];
    Min := auxInd[1];
    FOR I := 2 TO nter DO
        BEGIN
            FOR J := 1 TO nterminos DO
                Aux[J] := auxInd[I, J] - min[J];
{writeln('Aux=', StrInd(aux));}
            Normalizar(Aux);
            Contestacion := SeVerifica(Aux, MenorIgual);
            CASE Contestacion OF
                Si :
                    BEGIN
                        min := AuxInd[I];
                        mini := NN[I];
                    END;
                no :
                    BEGIN
                    END;
                NoSabe :
                    BEGIN
                        Mini0 := Mini;
                        Min0 := min;
                        IF iescritura THEN
                            writeln(StrInd(Aux), ' <= 0 ?');
                        IF automatico THEN
                            contestacion := no
                        ELSE
                            BEGIN
                                REPEAT
                                    writeln('Si restriccion cierta.....si, si falsa.....no');
                                    readln(contestacion);
                                UNTIL (contestacion <> nosabe);
                            END;
                        IF (Contestacion = si) OR automatico THEN
                            BEGIN
                                Mini := NN[I];
                                Min := auxInd[I];
                                GuardarenLista(Aux, MenorIgual);
                            END;
                        IF automatico THEN
                            BEGIN
                                casos[escribir] := casos[escribir] + 1;
                                IF iescritura THEN
                                    writeln('Escribe datos');
                                writedata(escribir);
                                NRestriccionesMenorIgual := NRestriccionesMenorIgual - 1;
                            END;
                        IF contestacion = no THEN
                            BEGIN
```

```
                        Mini := Mini0;
                        Min := Min0;
                        FOR J := 1 TO Nterminos DO
                            Aux[J] := -Aux[J];
                        GuardarenLista(Aux, Menor);
                    END;
                END;
            END;
        END;
    Minimo := Mini;
END;

FUNCTION anadelista (A : independiente) : boolean;
    VAR
        I : integer;
        anade : boolean;
BEGIN
    anade := true;
    I := 1;
    WHILE (I <= Nsoluciones) AND anade DO
        BEGIN
            IF Igual(A, soluciones[I, KK]) THEN
                anade := false;
            I := I + 1;
        END;
    IF anade THEN
        BEGIN
            Nsoluciones := Nsoluciones + 1;
            Soluciones[Nsoluciones, KK] := A;
        END;
    anadelista := anade;
END;

PROCEDURE InicializarDisco;

BEGIN
    drvNum := 1;
    leer := 1;
    escribir := 2;
    Filename[leer] := 'file1';
    Filename[escribir] := 'file2';
    Filename[3] := 'file3';
    Casos[leer] := 0;
    Casos[escribir] := 0;
    firsttime := true;
    err := GetVol(VolName, VrefNum[escribir]);
    VrefNum[leer] := VrefNum[escribir];
    VrefNum[3] := VrefNum[escribir];
    errorDisk(err);
    Err := FSOpen(Filename[escribir], VRefNum[escribir], RefNum[escribir]);
    ErrorDisk(err);
    Err := FSOpen(Filename[leer], VRefNum[leer], RefNum[leer]);
    ErrorDisk(err);
    Err := SetFPos(RefNum[escribir], FsFromStart, 0);
    ErrorDisk(err);
    Err := FSOpen(Filename[3], VRefNum[3], RefNum[3]);
    ErrorDisk(err);
    Err := SetFPos(RefNum[3], FsFromStart, 0);
    ErrorDisk(err);
    Err := SetFPos(RefNum[leer], FsFromStart, 0);
    ErrorDisk(err);
END;
```

```
PROCEDURE Inicializar;

   VAR
      I, J : integer;

BEGIN

   writeln('-------------------------------------------------');
   NRestriccionesInicialMenor := 0;
   NRestriccionesInicialMenorIgual := 0;
   N := 1;
   FOR I := 1 TO NumeroSucesos DO
      N := N * 2;
   M1 := N + 1;
   M2 := N + 3;
   Nterminos := N;
   FOR I := 1 TO M2 DO
      BEGIN
         FOR J := 1 TO M2 DO
            X[J, I] := 0.0;
         X[3, I] := 1.0;
      END;
   FOR I := 1 TO N DO
      x[I + 3, N - I + 1] := -1.0;
   FOR I := 1 TO M2 DO
      FOR J := 1 TO Nterminos DO
         Tindependiente[I, J] := 0.0;
   Tindependiente[3, 1] := 1.0;
END;

PROCEDURE escritura (A : TindependienteType;
                     xx : xtype);

   VAR
      i, j : integer;

BEGIN
   IF iescritura THEN
      BEGIN
         FOR I := 1 TO M2 DO
            BEGIN
               FOR j := 1 TO M1 - 1 DO
                  write(xx[i, j] : 5 : 1);
               write(' ');
               write(StrInd(A[I]));
               writeln(' ');
            END;
         writeln('---------------------------------------');
      END;
END;


PROCEDURE Formar_matriz_X;
   VAR
      I, J, JJ : integer;
      S : real;
BEGIN



   FOR I := 1 TO N DO
```

```
            X[1, I] := -C[I];

{Fila 2}

        FOR I := 1 TO N DO
            BEGIN
                S := 0.0;
                FOR J := 3 TO M + 3 DO
                    S := S + X[J, I];
                X[2, I] := S;
            END;

        FOR JJ := 1 TO Nterminos DO
            BEGIN
                S := 0.0;
                FOR J := 3 TO M + 3 DO
                    S := S + Tindependiente[J, JJ];
                Tindependiente[2, JJ] := S;
            END;
    END;

    PROCEDURE Simplex;


        VAR
            i, j, JJ : integer;
            valid : boolean;
            K, L : integer;
            pivot, S, auxil : real;
            aux : independiente;

        PROCEDURE comprobacion;

            VAR
                i, j : integer;

        BEGIN
            FOR I := 3 TO M2 DO
                BEGIN
                    FOR j := 1 TO M1 - 1 DO
                        IF NOT ((x[i, j] = 0.0) OR (x[i, j] = 1.0e+0) OR (x[i, j] = -1.0)) THEN
                            BEGIN
                                IF iescritura THEN
                                    writeln('error for i=', i : 2, ' j=', j : 2, '   x [ i , j ] =', x[i, j]);
                                error := true;
                            END;
                END;
        END;


    BEGIN

{Procedimiento SIMPLEX}


        REPEAT
            Done := false;
            REPEAT

{Calculo del mayor valor en fila M3}
```

```
                K := 1;
                S := X[M3, K];
                FOR J := 1 TO N DO
                    BEGIN
                        valid := (M3 = 2) OR (X[2, J] = 0.0);
                        IF (S < X[M3, J]) AND Valid THEN
                            BEGIN
                                S := X[M3, J];
                                K := J;
                            END;
                    END;
                IF iescritura THEN
                    BEGIN
                        writeln('Maximo valor=', S : 7 : 3, ' en columna ', K);
                        writeln('----------------------------------------');
                    END;
                IF (S <= 0.0) THEN
                    Done := true;
                IF Done = false THEN
                    BEGIN

{Cálculo del minimo X[I,N+1]/X[I,K]}
                        Nter := 0;
                        FOR I := 3 TO M + 3 DO
                            BEGIN
                                IF X[I, K] > 0.0 THEN
                                    BEGIN
                                        L := I;
                                        Nter := Nter + 1;
                                        IF iescritura THEN
                                            write('fila ', I : 3, ' ');
                                        FOR JJ := 1 TO Nterminos DO
                                            AuxInd[Nter, JJ] := Tindependiente[I, JJ] / X[I, K];
                                        NN[Nter] := I;
                                        IF iescritura THEN
                                            writeln(StrInd(AuxInd[Nter]));
                                    END;
                            END;
                        IF Nter > 1 THEN
                            BEGIN
                                L := Minimo;
                            END;
                        IF iescritura THEN
                            BEGIN
                                FOR JJ := 1 TO Nterminos DO
                                    Aux[JJ] := Tindependiente[L, JJ] / X[L, K];
                                IF iescritura THEN
                                    BEGIN
                                        writeln('Minimo valor=', StrInd(Aux), ' en fila    ', L);
                                        writeln('----------------------------------------');
                                    END;
                            END;

                        Auxil := X[L, K];
                        FOR J := 1 TO N DO
                            X[L, J] := X[L, J] / Auxil;
                        FOR JJ := 1 TO Nterminos DO
                            Tindependiente[L, JJ] := Tindependiente[L, JJ] / Auxil;

{Correccion de filas diferentes de L}
```

```
                    FOR I := 1 TO M + 3 DO
                        IF I <> L THEN
                            BEGIN
                                pivot := X[I, K] / X[L, K];
                                FOR J := 1 TO N DO
                                    X[I, J] := X[I, J] - X[L, J] * pivot;
                                FOR JJ := 1 TO Nterminos DO
                                    Tindependiente[I, JJ] := Tindependiente[I, JJ] - Tindependiente[L, JJ]
* pivot;
                            END;
                    escritura(Tindependiente, x);

                END;
            UNTIL Done;
            IF M3 = 2 THEN
                M3 := 1
            ELSE
                Done1 := true;
        UNTIL Done1;
        IF anadeLista(Tindependiente[1]) OR iescriturarestricciones THEN
            BEGIN
                IF bien THEN
                    BEGIN
                        IF KK = 1 THEN
                            BEGIN
                                write('El valor minimo es ');
                                Write(strInd(Tindependiente[1]));
                            END
                        ELSE IF KK = 2 THEN
                            BEGIN
                                write('El valor maximo es ');
                                FOR J := 1 TO Nterminos DO
                                    Tindependiente[1, J] := -Tindependiente[1, J];
                                Write(strInd(Tindependiente[1]));
                                FOR J := 1 TO Nterminos DO
                                    Tindependiente[1, J] := -Tindependiente[1, J];
                            END;
                    END
                ELSE
                    BEGIN
                        write('El valor minimo es ');
                        Write(strInd(Tindependiente[1]));
                    END;
                writeln('');
                IF iescriturarestricciones THEN
                    BEGIN
                        writeln(' si se verifican las restricciones');
                        writeln('');
                        writeRestricciones(NrestriccionesInicialMenorIgual + 1, NrestriccionesMenorIgual);
                    END;
            END;
    END;


BEGIN
    SetRect(TextR, 30, 40, 500, 300);
    SetTextRect(textR);
    HideAll;
    ShowText;
    writeln('Si automatico.....true, si no..... false');
    readln(automatico);
```

```
writeln('Si escritura.....true, si no..... false');
readln(iescritura);
writeln('Si escribir restricciones.....true, si no..... false');
readln(iescriturarestricciones);
writeln('Si nuevo.....true, si antes..... false');
readln(bien);
writeln(' Dar Numerosucesos');
readln(Numerosucesos);
InicializarDisco;
Inicializar;
SucesoNumber := N;
I9 := 0;
FOR K9 := 1 TO N - 1 DO
   BEGIN
      I9 := siguiente(I9);
      J2 := 0;
      FOR J88 := 1 TO N - 1 DO
         BEGIN
            J2 := siguiente(J2);
            IF BitAnd(I9, J2) = I9 THEN
               C[N - J88 + 1] := 1.0
            ELSE
               C[N - J88 + 1] := 0.0;
         END;
      C[1] := 0.0;
      M := N - SucesoNumber;
      NRestriccionesInicialMenorIgual := 0;
      M2 := M + 3;
      KK := 1;
      WHILE KK <= 2 DO
         BEGIN
            IF KK = 2 THEN
               BEGIN
                  FOR J2 := 1 TO N DO
                     C[J2] := -C[J2];
               END;
            NRestriccionesMenorIgual := NRestriccionesInicialMenorIgual;
            NRestriccionesMenor := NRestriccionesInicialMenor;
            M3 := 2;
            writeln('*****************************************');
            writeln('Suceso ', I9, ' caso =', KK);
            writeln('*****************************************');
            Nsoluciones := 0;
            leer := 1;
            escribir := 2;
            Filename[leer] := 'file1';
            Filename[escribir] := 'file2';
            Casos[leer] := 0;
            Casos[escribir] := 0;
            firsttime := true;
            Formar_matriz_X;
            escritura(Tindependiente, X);
            IF KK = 1 THEN
               BEGIN
                  Err := SetFPos(Refnum[3], FsFromStart, 0);
                  ErrorDisk(err);
                  writedata(3);
                  IF iescritura THEN
                     BEGIN
                        writeln('Matriz escrita en disco');
                        escritura(Tindependiente, X);
                     END;
```

```
                    END;
                Done1 := false;
                REPEAT
                    IF NOT firsttime THEN
                        BEGIN
                            Casos[leer] := Casos[leer] - 1;
                            IF iescritura THEN
                                writeln('lee datos');
                            readdata(leer);
                        END;
{ ================================}

                    Simplex;
{ ================================}
                    IF iescritura THEN
                        writeln('casos[leer]=', casos[leer], '  casos[escribir]=', casos[escribir]);
                    IF Casos[leer] = 0 THEN
                        BEGIN
                            Rebobinarall;
                            IF iescritura THEN
                                BEGIN
                                    writeln('Rebobinado realizado');
                                    writeln('casos[leer]=', casos[leer], '  casos[escribir]=', casos[escribir]);
                                END;
                        END;
                    firsttime := false;
                UNTIL Casos[leer] + Casos[escribir] = 0;
                Err := SetFPos(Refnum[3], FsFromStart, 0);
                ErrorDisk(err);
                readdata(3);
                IF iescritura THEN
                    BEGIN
                        writeln('Matriz leida de disco');
                        escritura(Tindependiente, X);
                    END;
                IF KK = 2 THEN
                    BEGIN
                        ManipularMatriz1;
                        sucesoNumber := sucesoNumber - 1;
                    END;
                KK := KK + 1;
            END;
        END;
    writeln('Por fin acabé');
END.
```

7.4.-LISTADO DEL PROGRAMA DE PROGRAMACIÓN PARAMÉTRICA

```
program LinPar2;


  const
    M1max = 33;
    M2max = 32;
    lowerextended = -9.0e+300;
    upperextended = 9.0e+300;

  type
    xtype = array[1..M2max, 1..M1max] of extended;
    vector = array[1..2] of extended;
  var
    textr: Rect;
    M1, M2: integer;
    M, N, M22, M23, I: integer;
    X: xtype;
    B: array[1..M2max, 1..M1max] of extended;
    C: array[1..M1max] of extended;
    done: boolean;
    Min, auxi, interval, interval1: vector;
    iescritura: boolean;
    f: text;
    Tindependiente: array[1..M2Max, 1..2] of extended;
    Str: string;
    VRefNum, refnum, PathRefNum: array[1..3] of integer;
    drvNum: integer;
    volName: stringPtr;
    leer, escribir, A: integer;
    Filename: array[1..3] of STR255;
    casos: array[1..2] of integer;
    firsttime: boolean;
    err: Oserr;
    Casosleidos, Casosescritos, Casosleermax: integer;
    Filenumber: integer;
    XinBasis, XRow: array[1..M1max] of integer;
    Neliminated: integer;
    failed: boolean;
    Solution, PartialSolution: extended;
    RowFree: array[1..M1max] of boolean;

  procedure Escritura;
    var
      I, J: integer;
  begin
    if iescritura then
      begin
        writeln(' ======= ', interval[1] : 12 : 4, ' ======= ', interval[2] : 12 : 4, ' ======= ');
        writeln(' ======= ', interval1[1] : 12 : 4, ' ======= ', interval1[2] : 12 : 4, ' ======= ');
        for J := 1 to M + 2 do
          begin
            for I := 1 to N do
              write(X[J, I] : 8 : 2);
            write(' =', Tindependiente[J, 1] : 8 : 2);
            if Tindependiente[J, 2] <> 0.0 then
              write(' +', Tindependiente[J, 2] : 8 : 2, ' A');
```

```
                writeln;
            end;
        writeln('------------------------------');
    end;
end;

procedure Errordisk (error: integer);

    var
        Dum: integer;

begin
    if (error <> noErr) and (error <> opWrErr) then
        begin
            Str := '';
            Str := Stringof(Str, error);
            ParamText('', Str, '', '');
            writeln('error=', Str);
            SysBeep(5);
            ExitToShell;
        end;
end;
procedure RebobinarAll;
    var
        Er: OSErr;
begin

    a := leer;
    leer := escribir;
    escribir := a;
    Er := SetFPos(RefNum[escribir], FsFromStart, 0);
    ErrorDisk(er);
    Er := SetFPos(RefNum[leer], FsFromStart, 0);
    ErrorDisk(er);
end;


procedure Writedata;


    var
        pt: point;
        TypeList: SFTypeList;
        reply: SFReply;
        I, J: integer;
        reloj: cursHandle;
        Temp: Str255;
        Error: integer;
        Dum: integer;
        Er: OSErr;

    procedure WriteByte (what: ptr; num: longInt);
        var
            error2: integer;
    begin
        error := FSWrite(refNum[fileNumber], num, what);
        if error <> noErr then
            begin
                error2 := FSClose(refNum[fileNumber]);
                ErrorDisk(error);
            end;
    end;
```

```
begin
   Filenumber := escribir;
   reloj := GetCursor(WatchCursor);
   SetCursor(reloj^^);
   if iescritura then
      begin
         writeln('Graba datos en disco');
         Escritura;
      end;
   WriteByte(@Done, sizeof(Done));
   WriteByte(@X, sizeOf(X));
   WriteByte(@Tindependiente, sizeOf(Tindependiente));
   WriteByte(@interval1, sizeOf(interval1));
   WriteByte(@Neliminated, Sizeof(Neliminated));
   WriteByte(@XRow, Sizeof(XRow));
   WriteByte(@XinBasis, Sizeof(XinBasis));
   WriteByte(@Rowfree, Sizeof(Rowfree));
   Casosescritos := Casosescritos + 1;
   SetCursor(Arrow);
end;

procedure readdata;


   var
      I, J: integer;
      reloj: cursHandle;
      Temp: Str255;
      Error: OsErr;
      MyDialog: DialogPtr;
      TheItem: integer;
      Dum: integer;
      L: longint;


   procedure ReadByte (what: ptr; num: longInt);
      var
         error2: integer;
   begin
      error := FSREad(refNum[fileNumber], num, what);
      if error <> noErr then
         begin
            error2 := FSClose(refNum[fileNumber]);
            ErrorDisk(error);
         end;
   end;

begin
   if Casosleidos = Casosleermax then
      begin
         Casosleidos := 0;
         Casosleermax := Casosescritos;
         Casosescritos := 0;
         RebobinarAll;
      end;
   if iescritura then
      begin
         writeln('Lee datos de disco');
      end;
   Filenumber := leer;
   ReadByte(@Done, sizeof(Done));
```

```
    ReadByte(@X, sizeOf(X));
    ReadByte(@Tindependiente, sizeOf(Tindependiente));
    ReadByte(@interval, sizeOf(interval));
    ReadByte(@Neliminated, Sizeof(Neliminated));
    ReadByte(@XRow, Sizeof(XRow));
    ReadByte(@XinBasis, Sizeof(XinBasis));
    ReadByte(@Rowfree, Sizeof(Rowfree));
    Casosleidos := Casosleidos + 1;
    Escritura;
end;


procedure InicializarDisco;

begin
    drvNum := 1;
    leer := 1;
    escribir := 2;
    Filename[leer] := 'file1';
    Filename[escribir] := 'file2';
    Filename[3] := 'file3';
    Casos[leer] := 0;
    Casos[escribir] := 0;
    err := GetVol(VolName, VrefNum[escribir]);
    VrefNum[leer] := VrefNum[escribir];
    VrefNum[3] := VrefNum[escribir];
    errorDisk(err);
    Err := FSOpen(Filename[escribir], VRefNum[escribir], RefNum[escribir]);
    ErrorDisk(err);
    Err := FSOpen(Filename[leer], VRefNum[leer], RefNum[leer]);
    ErrorDisk(err);
    Err := FSOpen(Filename[3], VRefNum[3], RefNum[3]);
    ErrorDisk(err);
    Err := SetFPos(RefNum[escribir], FsFromStart, 0);
    ErrorDisk(err);
    Err := SetFPos(RefNum[leer], FsFromStart, 0);
    ErrorDisk(err);
    Err := SetFPos(RefNum[3], FsFromStart, 0);
    ErrorDisk(err);

end;

function AlessB (var A, B: vector): boolean;
    var
        C: vector;
        Comp: extended;
begin
    AlessB := false;
    C[1] := A[1] - B[1];
    C[2] := A[2] - B[2];
    if C[2] = 0.0 then
        begin
            if A[1] < B[1] then
                begin
                    B := A;
                    AlessB := true;
                end;
        end
    else
        begin
            Comp := -C[1] / C[2];
            if C[2] < 0.0 then
```

```
            begin
               if Comp <= interval[1] then
                  begin
                     AlessB := true;
                  end
               else if Comp > interval[2] then
                  begin
                     AlessB := false;
                  end
               else
                  begin
                     AlessB := false;
                     interval1[1] := Comp;
                     interval1[2] := interval[2];
                     interval[2] := Comp;
                     WriteData;
                  end;
            end
         else
            begin
               if Comp >= interval[2] then
                  begin
                     AlessB := true;
                  end
               else if Comp < interval[1] then
                  begin
                     AlessB := false;
                  end
               else
                  begin
                     AlessB := false;
                     interval1[1] := interval[1];
                     interval1[2] := Comp;
                     interval[1] := Comp;
                     WriteData;
                  end;
            end
      end;
end;


function Intersection (var A, B: extended): boolean;
   var
      Comp: extended;
begin
   Intersection := false;
   if B = 0.0 then
      begin
         if A > 0.0 then
            begin
               Intersection := true;
            end;
      end
   else
      begin
         Comp := -A / B;
         if B < 0.0 then
            begin
               if (Comp < interval[1]) then
                  begin
                     Intersection := false;
                  end
```

```
                else if (Comp > interval[2]) then
                    begin
                        Intersection := true;
                    end
                else
                    begin
                        Intersection := true;
                        interval[2] := Comp;
                    end;
            end
        else
            begin
                if (Comp >= interval[2]) then
                    begin
                        Intersection := false;
                    end
                else if Comp < interval[1] then
                    begin
                        Intersection := true;
                    end
                else
                    begin
                        Intersection := true;
                        interval[1] := Comp;
                    end;
            end
    end;
end;


procedure GuardaRestriccion;
begin
end;

procedure ReadData0;

    var
        MM, I, J, JJ: integer;
        Sol: extended;
begin
    writeln('Numero del problema a leer ');
    readln(MM);
    for JJ := 1 to MM do
        begin
            read(f, N, M, Sol);
            if N <= 0 then
                Exittoshell;
            for I := 1 to N do
                read(f, C[I]);
            readln(f);
            C[N + 1] := 0.0;
            for J := 2 to M + 1 do
                begin
                    for I := 1 to N + 1 do
                        read(f, X[J, I]);
                    readln(f);
                end;
        end;
    writeln(N, ' variables and ', M, ' constraints', ' Sol=', Sol : 12 : 4);
    for I := 1 to N do
        write(C[I] : 12 : 4);
    writeln;
```

```
{Añadir Restricción}

        M := M + 1;
        for J := 1 to N + 1 do
            X[M + 1, J] := C[J];
        for J := 2 to M + 1 do
            begin
                for I := 1 to N + 1 do
                    write(X[J, I] : 12 : 4);
                writeln;
            end;
    end;


    procedure Formar_matriz_X;
        var
            I, J, JJ: integer;
            S: extended;
    begin


{Fila 1}

        for I := 1 to N + 1 do
            begin
                S := 0.0;
                for J := 2 to M + 1 do
                    S := S + X[J, I];
                X[1, I] := S;
                X[M + 2, I] := -C[I];
            end;

        for I := 1 to M + 2 do
            begin
                Tindependiente[I, 1] := X[I, N + 1];
                Tindependiente[I, 2] := 0.0;
            end;




    end;

    procedure Simplex;


        label
            1;

        var
            i, j, JJ: integer;
            valid: boolean;
            K, L: integer;
            pivot, S, auxil: extended;
            Comp1: extended;

    begin

{Procedimiento SIMPLEX}

        Done := false;
        repeat
```

```
{Calculo del mayor valor en fila 1}

            K := 1;
            S := X[1, K];
            for J := 1 to N do
                begin
                    if (S < X[1, J]) then
                        begin
                            S := X[1, J];
                            K := J;
                        end;
                end;

            if S <= 1.0e-8 then
                begin
                    Done := true;
                    writeln('End of Phase');
                end
            else if iescritura then
                begin
                    writeln('Maximo valor=', S : 7 : 3, ' en columna ', K);
                    writeln('----------------------------------------');
                end;
            if Done = false then
                begin

{Cálculo del minimo X[I,N+1]/X[I,K]}
                    L := M + 4;
                    for I := 2 to M + 1 do
                        if RowFree[I] then
                            begin
                                if X[I, K] > 0.0 then
                                    begin
                                        for J := 1 to 2 do
                                            auxi[J] := Tindependiente[I, J] / X[I, K];
                                        if L = M + 4 then
                                            begin
                                                L := I;
                                                Min := auxi;
                                            end
                                        else
                                            begin
                                                if AlessB(auxi, Min) then
                                                    begin
                                                        Min := auxi;
                                                        L := I;
                                                    end;

                                            end;
                                    end;
                            end;

                    if L = M + 4 then
                        begin
                            if X[M + 1, K] <= 0.0 then
                                writeln('Unbounded solution');
                            ReadData;
                        end;

                    if iescritura then
```

```
                    begin
                        writeln('Minimo cociente en fila ', L);
                        writeln('----------------------------------------');
                    end;


                Auxil := X[L, K];
                for J := 1 to N do
                    X[L, J] := X[L, J] / Auxil;
                Tindependiente[L, 1] := Tindependiente[L, 1] / Auxil;
                Tindependiente[L, 2] := Tindependiente[L, 2] / Auxil;

{Correccion de filas diferentes de L}


                for I := 1 to M + 2 do
                    if I <> L then
                        begin
                            pivot := X[I, K] / X[L, K];
                            for J := 1 to N do
                                X[I, J] := X[I, J] - X[L, J] * pivot;
                            Tindependiente[I, 1] := Tindependiente[I, 1] - Tindependiente[L, 1] * pivot;
                            Tindependiente[I, 2] := Tindependiente[I, 2] - Tindependiente[L, 2] * pivot;
                        end;
                    Escritura;
                    NEliminated := NEliminated + 1;
                    XinBasis[NEliminated] := K;
                    XRow[NEliminated] := L;
                    Rowfree[L] := false;
                end;


        until Done;
        if NEliminated = M then
            begin
                failed := false;
                I := 2;
                while (I <= M + 1) and not failed do
                    begin
                        if not Intersection(Tindependiente[I, 1], Tindependiente[I, 2]) then
                            failed := true;
                        I := I + 1;
                    end;
                if failed then
                    writeln('No solution by this way')
                else
                    begin
                        Partialsolution := interval[1];
                        writeln('partialsolution = ', PartialSolution : 12 : 4);
                    end;
            end
        else
            begin
                if (abs(Tindependiente[1, 2]) < 1.0e-10) then
                    begin
                        if (abs(Tindependiente[1, 1]) < 1.0e-10) then
                            begin
                                Partialsolution := interval[1];
                                writeln('partialsolution = ', Partialsolution : 12 : 4);
                            end
                        else if Tindependiente[1, 1] > 0.0 then
                            writeln('No feasible solution exists');
```

```
                end
            else
                begin
                    Comp1 := -Tindependiente[1, 1] / Tindependiente[1, 2];
                    if (Comp1 - interval[1] >= 1.0e-10) and (Comp1 - interval[2] <= 1.0e-10) then
                        begin
                            PartialSolution := Comp1;
                            writeln('partialsolution = ', PartialSolution : 12 : 4);
                        end;
                end;
        end;
    if PartialSolution < Solution then
        begin
            Solution := PartialSolution;
            for I := 1 to Neliminated do
                write(' X', Xinbasis[I] : 2, '=', Tindependiente[XRow[I], 1] + Tindependiente[XRow[I], 2]
* Solution : 12 : 4);
            writeln('');
        end;
1:
    writeln('=====================================');

end;


begin
    ShowText;
    InicializarDisco;
    iescritura := true;
    SetRect(TextR, 30, 40, 530, 300);
    SetTextRect(textR);
    Open(f, 'datoslineal');
    ReadData0;
    for I := 1 to M + 1 do
        RowFree[I] := true;
    Formar_matriz_X;
    Tindependiente[M + 1, 2] := 1.0;
    Tindependiente[1, 2] := 1.0;
    Solution := Upperextended;
    PartialSolution := Upperextended;
    casosleidos := 0;
    Casosescritos := 0;
    Casosleermax := 0;
    firsttime := true;
    leer := 2;
    escribir := 1;
    interval1[1] := 0.0;
    interval1[2] := upperextended;
    interval[1] := lowerextended;
    interval[2] := 0.0;
    NEliminated := 0;
    Escritura;
    writeData;
    for I := 1 to N + 1 do
        X[M + 1, I] := -X[M + 1, I];
    Formar_matriz_X;
    Tindependiente[M + 1, 2] := -1.0;
    Tindependiente[1, 2] := -1.0;
    Escritura;


{ ================================}
```

```
    repeat
        if not firsttime then
            begin
                ReadData;
            end;
        firsttime := false;
        Simplex;
        writeln('Casosleidos,Casosescritos=', Casosleidos, Casosescritos);
    until casosleermax - casosleidos + casosescritos = 0;
    writeln('*****************************************');
    if Solution = 9.0e+300 then
        writeln('Unfeasible solution')
    else if Solution = -9.0e+300 then
        writeln('Unbounded solution')
    else
        writeln('Solution=', Solution : 12 : 4);
    writeln('*****************************************');
{ ================================}
    err := FlushVol(VolName, VrefNum[escribir]);
    ErrorDisk(err);
end.
```

3 BIBLIOGRAFIA

BIBLIOGRAFÍA

8.-BIBLIOGRAFÍA

-**Abad, R. y Rey, G.** (1.984). Medicina e Informática. La informatización de la historia clínica. Editorial Médica Internacional S.A.

-**Abbott, J. C. (1969).** Sets, Lattice and Boolean Algebras. Allyn and Bacon. Boston.

-**Adams, J. B.** (1.976). A probability model of medical reasoning and the MYCIN model. Mathematical Biosciences , Vol. 32, 177-186.

-**Aikins, J. S., Kunz, J. C., Shortliffe, E. H. y Fallat, R. J.** (1.983). PUFF an expert sistem for interpretation of pulmonary function data. Comput. Biomed. Res., Vol. 16, 199-208.

-**Alello, N. y Nii, H. P.** (1.981). AGE-PUF:a simple event driven program. Computer Science Department Stanford University. Report nº HPP-81-25.

-**Alty, J. L. y Coombs, M.** (1.986). Sistemas Expertos : Conceptos y ejemplos : Métodos computacionales para la representación y el control. Diaz de Santos S.A. ed.

-**Angluin, D,** (1978). On the complexity of minimun inference of regular sets. Inf. Control 39. 337-350.

-**Angluin, D,** (1986). Learning regular sets from queries and counter-examples. Yale University. Tech. Rep. YALEU/DCS/464.

-**Armitage, P.** (1.971). Statistical methods in medical research. Blackwell Scientific publications.

-**Ashley, D. B. y Wharry, M. B.** (1985). Prototype expert system for subsurface risk. NSF Grant CEE-8352354.

-**Barnet, J. A.** (1.981). Computacional methods for a mathematical theory of evidence. In Proceedings of the 7th. International Joint Conference on Artificial Intelligence. Vancouver, 868-875. Vancouver, BG.

-**Barnwell, T. O., Jr., Brown, L. C. y Marek, W.** (1986). Development of a prototype expert advisor for the enhanced stream water quality model QUAL2E. Internal Report, U.S. Environmental Protection Agency, Environmental Research Laboratory. Athens, GA.

-**Barr, A., Feigenbaum, E. A. y Cohen, P. R.** (1.981). The handbook of Artificial Intelligence (3 Vol.), Willians Kaufman. Los Altos Inc.

-**Bennet, J. S., Creary, L., Engelmore y R., Melosh, R.** (1.978). SACON. A knowledge-based consultant for structural analysis. Computer Science Department. Stanford University. Report HPP78-23.

-**Bishop, Y. M. M., Fienberg, S. A. y Holland, P. W.** (1975). Discrete multivariate analysis: Theory and Practice. Cambridge, Mass., The MIT Press.

-**Bowen, J., Cornick, T. C. y Bull, S. P.** (1986). BERT- An expert system for brickwork design. Working paper, University of Reading, England, Departments of Computer Science and Construction Management.

-**Brachman, R. J.** (1.976). What's in a concept : Structural foundations for semantic networks. Bolt Beraneck and Newman. Report 3433.
-**Bratnagar, R. K. y Kanal, L.N.** (1986). Handling uncertain information: a review of numeric and non-numeric methods. In Uncertainty in Artificial Intelligence, 3-34. L. N. Kanal and J. F. Lemmer ed. Elsevier Science Publishers.
-**Buchanan, B. G. y Duda, R. O.** (1.983). Principles of rule-based expert systems. Advances in Computers, Vol. 22, 164-216. New-York Academic Press.
-**Buchanan, B. G. y Feigenbaum, E. A.** (1.978). DENDRAL and meta-DENDRAL. Their applications dimension. Artificial Intelligence, Vol . 1, 5-24.
-**Buchanan, B. G. y Mitchel, T.** (1.978). Model directed learning of productions rules. In Patern-Directed Inference Systems, 297-312. D.A.Waterman and F.Hayes-Roth ed. Academic Press. New-York.
-**Buchanan, B. G. y Shortliffe, E. H.** (1.984). Rule-based expert systems : The MYCIN experiments of the Stanford heuristic programming project. Reading, Mass. Addison Wesley Publissing Company.
-**Buchanan, B. G., Smith, D. H., White, W. C., Gritter, R. F., Feigenbaun, E. A., Lederberg, J. y Djerassi, C.** (1.976). Aplications of Artificial Intelligence for chemical inference. Automatic rule formation in mass spectrometry by means of the meta-DENDRAL program. Journal of the American Chemical Society, Vol. 96, 61-68.
-**Buchanan, B. G., Sutherly, G. L. y Feigenbaum, E. A.** (1.969). Heuristic DENDRAL: a program for generating explanatory in organic chemistry. In B. Meltzer and D.Michel ed. Machine Intelligence, Vol. 4. Edimburg University Press.
-**Buchanan, B. G., Sutherly, G. L. y Feigenbaum, E. A.** (1.970). Rediscovering some problems of Artificial Intelligence in the context of organic chemistry. In B.Meltzer and D.Michel ed. Machine Intelligence, Vol. 5, 209-254. Edimburg University Press.
-**Burger, R. y Fischer, M.** (1985). An expert system for project organization, presented to INTERNET Seminar.
-**Cagan, J. y Genberg, V.** (1987). PLASHTRAN, an expert consultant on two-dimensional finite element modelling techniques. Engineering with Computers.
-**Camacho, G.** (1985). LOW-RISE: An expert system for structural planning and design of industrial buildings. Master's thesis, Deparment of Civil Engineering, Carnegie-Mellon University, Pittsburgh, PA.
-**Carnap, R.** (1.962). The aim of inductive logic. In Logic Methodology and Philospy of Science, 303-318. Negel Supper and tarskieds. Standford. A Standford University Press.
-**Carnap,R.** (1.950). The two concepts of probability. In Logical Foundations of probability, 19-51. Chicago. University of Chicago Press.
-**Castillo, E.** (1.978). Introducción a la Estadística aplicada. Ed. E. Castillo. Santander. Spain.

-**Castillo, E.** (1.982). Bioestadística. Acta Otorrinolaringológica Española, Vol 35 (5), 777-808.
-**Castillo, E. y Alvarez, E.** (1.987). Some probabilistic environments in expert systems. Technical Report. Universidad de Cantabria.
-**Castillo, E., Alvarez, E., Zaballa, G. y Naranjo, A.** (1.988). A shell structure of probabilistic type for expert systems. Systémes expertes et leur applications. Avignon.
-**Castillo, E. y Alvarez, E.** (1.988). Entorno probabilístico para utilización de sistemas expertos. Jornadas matemáticas hispano-lusas. Valladolid.
-**Castillo, E., Luceño, A., Mora, E. y Puig-Pey, J.** (1.980). Bioestadística. Universidad de Cantabria. Spain
-**Chandrassekaran, B., Conus, F., Mitale, S. y Smith, J.** (1.979). An approach to medical diagnosis based on conceptual schemes. In proceedings of the 6th International Joint Conference on Artificial Intelligence, 134-142. Tokio.
-**Charniak, E.** (1.977). A frame painting:The representations of a common sense knowledge fragment. Journal of Cognitive Science, Vol. 1 (4), 355-394.
-**Charniack, E. y Mac Dermot, D.** (1.985). Introduction to Artificial Intelligence. Addison-Wesley Publishing Company.
-**Chatalic, P., Dubois, D. y Prade, H.** (1987). An approach to approximate reasoning based on the Dempster rule of combination. International Journal of Expert Systems, Vol. 1, No 1, 67-87.
-**Cheeseman, P.** (1.985). In Defense of probablity. International Joint Conference in Artificial Intelligence, 1002-1009.
-**Clancey, W. J.** (1.979). Dialogue management for rule-based tutorials. Proceedings of the 6th. International Joint Conference on Artificial Intelligence, 155-161. Tokyo.
-**Clancey, W. J.** (1.983). The epistemology of a rule-based expert sistem : a framework for explanation. GUIDON. Artificial Intelligence, Vol. 20, 215-251.
-**Clark, K. L. y Mc.Cabe, F. G.** (1.982). PROLOG : A language for implementing expert systems. J.Hayes, D.Michie and Y.Pao ed. Machine Intelligence, 455-470. John Wiley. New-York.
-**Cohn, L., Harris, R. y Bowlby, W.** (1986). Using expert systems for transportation noise decision making. Transportation Policy and Decision Making, Martinum Nijhoff, Dordrecht. The Netherlands.
-**Cuena, J.** (1.987). Lógica informática. El razonamiento aproximado en sistemas expertos. Alianza Editorial S.A. Madrid.
-**Cumberbach, Leung, V. K. y Heaps,H. S.** (1.974). A non-probabilistic method for automated medical diagnosis. International Journal of Biomedical Computing, Vol. 5, 133-146.
-**Darroch, J. N., Lauritzen, S. L. y Speed, T. P.** (1980). Markov fields and log-linear interaction models for contingency tables. The Annals of Statistics, Vol. 8, No. 3, 522-539.

-**Davis, R.** (1.977). Interactive transfer of expertice. Adquisition of a new Inference rules. In Proceedings of the 5th. International Joint Conference on Artificial Intelligence, 321-328. Cambrigde MA.

-**Davis, R.** (1.979). Interactive transfer of expertise:Adquisition of new inference rules. Artificial intelligence, Vol 12, 121-158.

-**Davis, R.** (1.980). Meta rules: Reasoning about control. Artificial Intelligence, Vol. 15, 179-222.

-**Davis, R.** (1.983). TEIRESIAS: Experiments in communications with a knowledge-based expert sistem. In M.E. Coombs ed. Designing for human computer comunication. Academic Press. London.

-**Davis, R. y Buchanan,** B. G. (1.977). Meta level knowledge. Overview and applications. In proceedings of the 5th International Joint Conference on Artificial Intelligence, 920-927. Cambrigde MA.

-**Davis, R., Buchanan,** B. y Shortliffe, E. (1.977). Production rules as a representation for a knowledge-based consultation system. Artificial Intelligence, Vol. 8 (1), 15-45.

-**Day, E.** (1.976). Automated health services: Reprogramming the doctor. Methods of Information in Medicine, Vol 9, 116-121.

-**DeDombal, F, T., Haper, D. J., Horrocks, J. C. y Mc.Cann, A. P.** (1.974). Human and computer aided diagnosis of abdominal pain : Further report with emphasis on performance of clinicians. British Medical Journal, Vol 2, 376-380.

-**DeDombal, F.T., Horrocks, J. C. y Staneily, J. R.** (1.975). The computer as an aid to gastroenterologycal decission making.

-**Diekmann, J. E. y Kruppenbacher, T. A.** (1984). Claims analysis and computer reasoning. Journal of Construction Engineering and Management, Vol. 110, No. 4, 391-408.

-**Dietterich, T. C. y Michalski,** R. S. (1983). A comparative review of selected methods for learning from examples. Machine Learning: An Artificial Appoach. Tioga, Palo Alto, Calif.

-**Domenech, J. M.** (1.977). Bioestadística. Ed. Herder. Barcelona. Spain .

-**Dubois, D. y Prade, H.** (1987). Una approche ensembliste de la combinaison d'informations imprécises ou incertaines. Revue d'intelligence artificielle, Vol. 1, No 4, 23-42.

-**Duda, R. O.,Hart, P. E. y Nilson,** N. (1.976). Subjetive bayesian methods for rule based inference systems. Proceedings of the National Computer Conference AFIPS, Vol. 45, 1.075-1082.

-**Duda, R. O., Hart, P. E., Nilson, N. y Shutherly, G. L.** (1.978a). Semantic network representations. In rule-based inference systems, 203-221. Pattern Directed Inference sistems. D.A.Waterman and F. Hayes-Roth. Academic Press. New York.

-**Duda, R. O., Gaschning, J. y J.Hart, P. E.** (1.979). Model desing in the PROSPECTOR consultant program for mineral exploration. D.Michie ed.. Expert Systems in the Microelectronic Age. Edimburg. Edimburg University Press .

-**Duda, R. O., Gaschning, J., Hart, A. E., Konolige, K., Barret, P. y Slocum, J.** (1.978b). Development of the PROSPECTOR. Consultation system for mineral explorations. Final Report.S.R.I, 5821-6415. Projects. S.R.I. International Inc. Menlo Park. C.A.

-**Duda, R. O. y Shortliffe, E. H.** (1.983). Expert-systems research. Science, Vol 220, 261-268.

-**Edwards, F. H., Davies, R. S.** (1.984). Use of bayesian algorithm in the computer assisted diagnosis of appendicitis. Sur. Gynecol. Obstet., Vol. 158, 219-222.

-**Engman, E. T., Rango, A., y Martinec, J.** (1986). An expert system for snowmelt runoff modeling and forecasting. Water Forum´86, 174-180. ASCE, New York.

-**Erman, L. D., Hayes Roth, I., Lesser, V. R. y Reddy, D. R.** (1.980). The HERSAY II speech understanding systems. Integration knowledge to resolve uncertainty. Computers Surveys, Vol. 12, 13-253.

-**Fayegh, D. y Russell, S. O.** (1986). An expert system for flood estimation. In Expert Systems in Civil Engineering, 174-181. ASCE. New York.

-**Feigenbaum, E. A., Buchanan, B. G. y Lederbeg, J.** (1.971). One generality and problem solving : A case study involving the DENDRAL program. Machine Intelligence, Vol. 6, 165-190.

-**Feltovich, P.J., Johnson, P.E., Moller, J.H. y Swanson, D.** (1.980). The role on development of medical knowledge in diagnosis expertice. Paper presented at the annual meeting of the American Educational Research Association.

-**Fenves, S. J., y Maher, M. L.** (1981). The use of Artificial Intelligence techiques in preliminary structural design. Technical Report, Design Research Center Report. Carnegie Mellon University, Pittsburgh, PA.

-**Feurzeig, W., Munter, P., Swets, J. y Breen,M.** (1.964). Computer aided teaching in medical diagnosis. Journal of Medicine Education, Vol. 39, 746-755.

-**Fienberg, S. E.** (1977) The analysis of cross-classified categorical data.

-**Finn, G. y Reinschmidt, K. F.** (1986). Expert systems in an engineering construction firm. Expert Systems in Civil Engineering. ASCE. New York.

-**Forgy, C. y Mac Dermott, J.** (1.975). OPS: A domain-independent production system language. International Joint Conference in Artificial Intelligence, Vol. 5, 933-939.

-**Forsyth, R. y Rada, R.** (1986). Machine learning : Applications in expert systems and information retrieval. Halsted Press, a Division of John Wiley and Sons.

-**Fred, H., Edwards, M. S. y Geoffrey, M.** (1.987). The theorem of Bayes as a clinical research tool. Surg. Ginecol. Obstet., Vol 165, 127-129.
-**Friedman, L.** (1.971). Extended plausible inference. Proceeding of the 7th. International Joint Conference on Artificial Intelligence, 487-495. Vancouver BC.
-**Frost, R.** (1986). Introduction to knowledge base systems. Collins professional and technical books, William Collins Sons and Co., Ltd.
-**Fu, K.S. y Yao,J.T.P.** (1984). SPERIL: An expert system for safety evaluation of structures. In Proceedings of the IEEE Conference on Systems, Man and Cybernetics.
-**Furuta, H., King-Sun, T. y Yao, J. T. P.** (1985). Structural engineering applications of expert systems. Computer Aided Design, Vol. 17(9), 410-419.
-**Gaines, B. R.** (1986). Foundations of knowledge engineering. Proceedings of the Sixth Technical Conference of the British Computer Society Specialist Group on Expert Systems. Brighton.
-**Galambos, J.** (1987). The asymptotic theory of extreme order statistics. Robert E. Krienger. Publishing Company, Malabar, Florida.
-**Garrett, J. H.** (1986) A knowledge-based standards processor for structural component design. Thesis presented to Carnegie-Mellon University in partial fulfillment of requirements for the degree of Doctor of Philosophy. Pittsburgh, PA.
-**Garvey, T. D., Lowrence, J. D. y Fischler, M. A.** (1.981). An inference technique for integrating knowledge from disparate sources. Proceedings of the 7th. International Joint Conference on Artificial Intelligence, 319-325. Vancouver BC.
-**Gaschning, J.** (1.979). Preliminary performance analysis of the PROSPECTOR consultant system for mineral exploration. Proceedings of the 6th International Joint Conference in Artificial Intelligence, 308-310. Tokyo.
-**Gass, S.I.** (1969). Linear Programming. Methods and Applications. Mc. Graw Hill Book Co. New York.
-**Gero, J. S., y Coyne, R. D.** (1986). Developments in expert systems for design synthesis. Expert Systems in Civil Engineering. ASCE. New York.
-**Gill, A.** (1976). Appied algebra for the computer sciences. Prentice-Hall. Series in automatic computation.
-**Glesser, M. A. y Collen, M.F.** (1.972). Towards automated medical decisions. Computers and Biomedical Research, Vol. 5, 180-189.
-**Goodall, A.** (1985). The guide to expert systems. Learned Information Ltd.
-**Gorry, G. A.** (1.973). Computer assisted clinical decision making. Methods of Information in Medicine, Vol. 12, 45-51.
-**Gray, C. y Little, J.** (1985). A systematic approach to the selection of an appropiate crane for a construction site. Construction Management and Economics, Vol. 3, 121-144.

-**Gray, C. y Little, J.** (1986). Expert system development for predicting time and cost of construction during Initial design. First International Expert Systems Conference. London.
-**Grayson, C. J.** (1.960). Decision under uncertainty : Drilling decisions by gas and oil operators. Cambrigde MA. Harvard University Press.
-**Gregory, B. L., y Shephard, M. S.** (1986). Design of a knowledge based system to convert airframe geometric models to structural models. Expert Systems in Civil Engineering. ASCE. New York.
-**Gupta, M. M., Saridis, G. N. y Gaines, B. R.** (1977). Fuzzy Automata and Decision Processes. North-Holland. New York.
-**Haas, C.** (1986). Preserver: A pavement maintenance consultant. Technical Report, Department of Civil Engineering. Carnegie Mellon University. Pittsburgh, PA.
-**Hadden, W. J., Jr. y Hadden, S. G.** (1985). Expert systems for environmental regulation. In Expert Systems in Government (Symposium) 558-566. IEEE Computer Society.
-**Hajek, J. J., Chong, G. J., Haas, R. C. G. y Phang, W. A.** (1987) Knowledge-based expert system technology can benefit pavement maintenance. Presented at the 66th Annual TRB Meeting.
-**Halmos, P. R.** (1967). Lectures on boolean algebras. Princeton, NJ: Van Nostrand.
-**Harmon, P. y King, D.** (1.985). Representing knowledge John Willey and Sons. I.N.C.A Willey Pres Book.
-**Harré, R.** (1.970). Probability and confirmation in the principles of scientific thinking. University of Chicago Press.
-**Harrys, R., Cohn, L. y Bowlby, W.** (1985). An application of Artificial Intelligence in highway noise analysis. Transportation Research Record 1033, TRB. National Academy of Sciences. Washington, D.C.
-**Harrys, R., Cohn, L. y Bowlby, W.** (1987). Designing noise barriers using the expert system CHINA. Journal of Transportation Engineering. ASCE, Vol. 113, No.2, Proc. Paper 21310.
-**Hart, A.** (1986). Knowledge adquisition for expert systems. Kogan Page Ltd.
-**Heckerman, D.** (1986). Probabilistic interpretation for MYCIN's certainty factors. In Uncertainty in Artificial Intelligence, 177-201. L. N. Kanal and J. F. Lemmer ed. Elsevier Science Publishers.
-**Hempel, C. G.** (1.965). Studies in the logic of confirmation Aspects of Scientific Explanation and other Essays in the Philosophy of Science, 3-51. New-York Free Press.
-**Hendrickson, C., Martinelli, D. y Rehak, D.** (1986). Hierarchical rule-based activity duration estimation. Working paper. Department of Civil Engineering, Carnegie Mellon University.

-**Hendrix, G. G.** (1.976). The lifer manual : A guide to building practical natural language interfaces. Artificial Intelligence Center Stanford Reseach Institute. Report 138.

-**Hendrix, G. G.** (1.977). A natural language interface facility. Sigart Newsletter, Vol. 61, 25-26.

-**Hendrix, G.** (1.979). Encoding knowledge in partitional networks. In N. V. Findler ed. Associative Networks. Representation and use of knowledge by computers. London Academic Press.

-**Hii, H. P. y Feigenbaum, E. A.** (1.978). Rule-based understanding of signal in pattern-directed inference systems. D.A. Waterman and F.Hayes-Roth ed. Academic Press. New York.

-**Holmblad, L. P.** (1987). Automatic control of cement kiln by fuzzy logic techniques. Seminar on Expert Systems and their Applications. Santander, Spain.

-**Howard, C.** (1983). HICOST.Technical Report, Project Report 12-743, Expert Systems in Civil Engineering, Deparment of Civil Engineering. Carnegie Mellon University, Pittsburgh PA.

-**Howard,H.C.** (1986). Interfacing databases and knowledge-based systems for structural engineering applications, Thesis presented to Carnegie Mellon University, in Pittsburgh, PA, in partial fulfillment of requirements for the degree of Doctor of Philosophy.

-**Huang, M. S., Shenoi, S., Mathews, A. P., Lai, F. S. y Fan, L. T.** (1986). Faulty diagnosis of hazardous waste incineration facilities using a fuzzy expert system. In Expert Systems in Civil Engineering, 30-37. C.N.Kostem and M.L.Maher ed. ASCE. New York.

-**Hunt, E. B., Marin, J. y Stone, P. T.** (1966). Experiments in induction. Academic Press.

-**Hushon, J. M.** (1986). Response to chemical emergencies. Environmental Science and Technology, Vol. 20(2), 118-121.

-**Hushon, J. M.** (1986). Expert system for first responders to chemical emergencies. In ACS Annual Meeting.

-**Hutchinson, P.** (1985). An expert system for the selection of earth retaining structures. Master's thesis, Department of Architectural Science. University of Sydney. Australia.

-**Ishizuka, M., Eu, K. S., y Yao J. T. P.** (1982). SPERIL: An expert system for damage assessment of existing structures. In Proceedings of the 6th International Conference on Pattern Recognition, 832-937. Institute of Electrical and Electronics Engineers.

-**Karakatsanis, A.** (1985) FLODER: An expert systems for floor plan layout. Master's thesis. Department of Civil Engineering, Carnegie Mellon University, Pittsburgh, PA.

-**Kareem, A., y Allen, R. H.** (1987). WISER: A knowledge-based expert system for the design modification of high-rise buildings for serviceability. Journal of Wrel Engineering and Industrial Aerodynamics.
-**Klahr, P. y Waterman, D. A.** (1986). Expert system techniques, tools and applications. Addison Wesley Publishing Co.
-**Klir, G. J. y Folger, T. A.** (1989). Fuzzy sets, uncertainty and information. Prentice-Hall International Editions.
-**Konkoly, G. M.** (1986). A shallow trench design expert system. Master's thesis. Carnegie Mellon University, Pittsburgh, PA.
-**Kostem, C. N.** (1986). Design of an expert system for the rating of highway bridges. Expert Systems in Civil Engineering, ASCE. New York.
-**Kulikoski, C.** (1.982). Representation of expert knowledge for consultation : The CASNET and EXPERT projects. Artificial Intelligence in Medicine, 21-25. Szolovits ed. Westview Press.
-**Kulkoski, C. y Wels, S.** (1.971). Computer-based models of glaucoma. Department of Computer Science. Rutgers University. Computers in Biomedicine. Report nº 3.
-**Kunz, J. C., Bonura, T., Stelzner, M. J. y Levitt, R. E.** (1986). Contingent analysis for project management using multiple worlds. Applications of Artificial Intelligence to Engineering Problems, Vol. II.
-**Kunz, J. C., Shortliffe, E. H., Buchanan, B. G. y Feigenbaum, E. A.** (1.984). Computer assisted decision making in Medicine. The Journal of Medicine an Philosophye, Vol. 9, 135-160.
-**Lachenbruch, P. A.** (1975). Discriminant Analysis. Hafner Press, MacMillan Publishing Co., Inc. New York, London.
-**Langlotz, C. P. y Shortliffe, E. H.** (1.983) .ONCOCIN : Adapting a consultation system to critique use plans. International Journal of Man-Machine Studies, Vol. 19, 479-496.
-**Law, K. H., Zimmie,T. F. y Chapman, D. R.** (1986). An expert system for inactive hazardous waste site characterization. In Expert Systems in Civil Engineering, 159-170. C.N.Kostem and M.L. Maher ed. ASCE. New York.
-**Lesser, R. L., Fennel, R. D., Erman, L. D. y Reddy, D.** (1.974). Organization of the HERSAY II speech understanding systems. Contributed papers of the I.E.E.E. Symposium on speech recognition, 11-21. Pittsburg.P.A.
-**Lesser, R. L.,Fennel, R. D., Erman, L. D. y Reddy, D.** (1.975). Organitation of the HERSAY II speech understanding systems. I.E.E.E. Symposium. Transaction on acustic speech and signal processing ASPP, Vol. 23, 11-23.
-**Levitt, R. E.** (1986). HOWSAFE : A microcomputer-based expert system to evaluate the safety of a construction firm. In Expert Systems in Civil Engineering, C.Kostem and M.L. Maher, ed. ASCE.

-**Levitt, R. E.,** Samelson, N. M. y Parker, **H. W.** (1981). Improving Construction safety performance: The user's role. Technical Report # 260, Department of Civil Engineering. Standford University.
-**Lindley, D. V.** (1.987). The probability approach to the treatment of uncertainty in Artificial Intelligence and expert systems. Statistical Science, Vol. 2, No. 1, 17-24.
-**Lindsay, R. K., Buchanan, B. G., Feigenbaum, E. A. y Lederberg, J.** (1.980). Applications of Artificial Intelligence for Organic Chemistry. The DENDRAL Project. Mc Graw-Hill. New-York.
-**Ludvigsen, P. J., Simms, R. C. y Grenney,W .J.** (1986). Development of a prototype expert for assessing organic chemical mobility and degradation to determine soil treatment requirements. In Fourth Conference on Computing in Civil Engineering, ASCE.
-**Luce, R. D. y Suppers, P.** (1.965). Preference utility and subjective probability. In Handbook of Mathematical psychology, 249-410. Ed. R.D.Luce, R.R.Bush and E. Gallanter. Wiley. New york.
-**Luceño, A.** (1988). Métodos de Estadística aplicada. Servicio de publicaciones. Universidad de Cantabria.
-**Mac Dermott, J.** R-1: A ruled-based configurer of computer systems. Carnegie Mellon University. Report CMU.CS, 80-119.
-**Mac Dermot D.V. y Doyle, J.** (1.980). Non monotonic logic. Artificial Intelligence, Vol. 13, 41-72.
-**MacLane, S. y Birkhoff, G.** (1967). Algebra. Macmillan. New York.
-**Mc Carthey, J.** (1.962). History of LIPS. Siglan Notices, Vol. 13, 117-223. Reprinted in the Handbook of Artificial Intelligence (Vol. 2).
-**McNeil, S. y Finn, A.** (1987) . An expert system to cost feasible bridge painting strategies (Bridge PIARS). Transportation Research Record. National Academy of Science. Washington, D. C.
-**Maher, M.L.** (1986). Problem solving using expert system techniques. Expert Systems in Civil Engineering, 7-17. ASCE. New York.
-**Menger, k.** (1942). Statistical metrics. Proc. Nat. Acad, Sci., 28, 535-537.
-**Michalski, R.** (1983). Unifying principles and a methodology of inductive learning. Artificial Intelligence.
-**Michalski, R y Chilausky, R.** (1980). Learning by being told and learning from examples. An experimental comparison of the two methods of knowledge acquisition. Policy analysis and Information systems. Junio.
-**Michalsky, R., Carbonell, R y Mitchell, T.** (1982). Machine learning. Ed. Tioga.
-**Mikroudis, G. K., Fang, H. Y. y Wilson, J. L.** (1986). Development of GEOTOX expert system for assessment of hazardous waste sites. In 1st International Symposium on Environmental Geotechnology. Lehigh University.

-**Miller, R. A., Pople, H. E. y Myers, J. D.** (1.982). INTERNIST-I a experimental computer based diagnosis consultant for general internal medicine. New England Journal of Medicine, Vol. 307(8), 468-476.

-**Minski, M. L.** (1.975). A frame-work for representing knowledge. In the Psychology of Computer Vision, 221-277. PH. Wiston ed. Mc Graw-Hill. New-York.

-**Miyasato, G. H., et. al.** (1986). Implementation of a knowledge-based seismic risk evaluation system on microcomputers. International Journal for Artificial Intelligence in Engineering.

-**Mullarkey, P. W.** (1985). CONE : An expert system for interpretation of geotechnical characterization data from cone penetrometers. Thesis presented to Carnegie Mellon University, at Pittsburgh, PA, in partial fulfillment of requirements for the degree of Doctor of Philosophy.

-**Mulsant, B. y Servan-Schreiber, D.** (1.984). Knowledge engineering : A daily activity on a hospital ward. Computers and Biomedical Reseach, Vol. 17, 71-79.

-**Myers, J. D., Pople, H. E. y Miller, R. A.** (1.982). INTERNIST. Can artificial Intelligence help. Connelly, Benson, Burke and Fenderson ed. Clinical Decisions an Laboratory use. Minneapolis. University of Minnesota Press.

-**Naranjo, A.** (1988). Aplicación de los sistemas expertos al diagnóstico médico y a la enseñanza de la Medicina. Tesis doctoral. Universidad de Cantabria. Spain.

-**Naylor, C. H.** (1.986). Construya su propio sistema experto. Diaz de Santos ed.

-**Newel, A. y Simon, H. A.** (1.972). Human problem solving. Englewood Cliffs,N.J.

-**Niwa, K. y Okuma, M.** (1982). Know-how transfer method and its application to risk management for large construction projects, IEEE Transactions on Engineering Managements, Vol. EM- 29, No. 4.

-**O´Connor, M. J., De la Garza, J. M. y Ibbs, C. W.** (1986). An expert system for construction schedule analysis. In Expert Systems in Civil Engineering. C.Kostem and M.L. Maher ed. ASCE.

-**Parzen, E.** (1.960). Modern probability theory and its applications. Wiley. New-York.

-**Pauker, S. G., Gorry, G. A., Kassirer, J, P. y Schwartz, W. B.** (1.976). Towards the simulation of clinical cognition.The American Journal of Medicine, Vol. 60, 981-996.

-**Paulson, B. C., Jr. y Sotoodeh-Khoo, H.** (1987). Expert systems in real-time construction operations. Proceedings of CIB W-65 Symposium. London, England.

-**Paquette, J. S. Woodson, L. y Bissex, D. A.** (1986). Improving the implementation of remedial investigation. Feasibility studies using computerized expert systems. In Superfund´86, Hazardous Materials Control Research Institute.

- **Pitt, L. and Valiant, L. G.** (1988). Computational Limitations on Learning from Examples. Journal of the Association for Computing Machinery, Vol 35, No. 4, 965-984.
-**Poper, K. R.** (1.959). Corroborating the weigth of evidence. In the Logic of Scientific Discovery, 387-419. Scientific Editions. New-York.
-**Quinlan, J. R.** (1979). Discovering rules from large collections of examples: a case study. Formando parte de Expert Systems in the Micro electronic age. Edinburgh University Press.
-**Rao, C. R.** (1973). Linear statistical inference and its applications. John Wiley. New York.
-**Reddy, D. R., Hernan, L. D., Fennel, R. D. y Neely, R. B.** (1.973). The HERSAY speech understanding systems: An example of the recognition process. Avances papers of the 3rd International Joint Conference on Artificial Intelligence, 185-193. Stanford CA.
-**Rehak, D., Howard, H. C. y Sriram, D.** (1985). Architecture of an integrated knowledge based environment for structural engineering applications. Knowledge Engineering in Computer Aided Design, North-Holland Publishing Co.
-**Reichgel, H. y van Harmelen, F.** (1985). Relevant criteria for choosing an inference engine in expert systems. Proceedings of the Fifth Technical Conference of the British Computer Society Specialist Group on Expert Systems, 21-30.
-**Ritchie,S.** (1987). Expert systems in pavement management. Transportation Research, Part A.
-**Ritchie, S., Yeh, C., Mahoney, J. y Jackson, N.** (1986). Development of an expert system for pavement rehabilitation decision-making. Transportation Research Record 1077, TRB, 96-103. National Academy of Science. Washington, D.C.
-**Ritchie, S., Yeh, C., Mahoney, J. y Jackson, N.** (1987). A surface condition expert system for pavement rehabilitation planning. Journal of Transportation Engineering, Vol. 113, No. 2, Proc. Paper 21367. ASCE.
-**Rooney, M. F.** (1986). Expert systems in structural engineering. Survey of the state of the art expert knowledge based systems in Civil Engineering, USA-CERL Special Report P 87/01.
-**Rosenman, M. A. y Gero, J. S.** (1985). Design codes as expert systems. Computer-Aided Design, Vol. 17(9).
-**Rosenman, M. A., Gero, J. S. y Oxman, R.** (1986). An expert system for design codes and design rules. Applications of Artificial Intelligence to Engineering Problems.
-**Rosemberg, S.** (1.977). Frame-based text processing. Artificial Intelligence Laboratory Massachusetts Institute of Technology. Report nº 431.

-**Ross, T. J. y Wong, F. S.** (1986). Structural damage assessment using AI techniques. Applications of Artificial Intelligence to Engineering Problems.
-**Rossman, L. A. y Haxo, H. E., Jr.** (1985). A rule-based inference system for liner/waste compatibility. In Environmental Engineering Specialty Conference, 588-590. ASCE.New York.
-**Sachdeva, P.** (1985). DAMP-A : Diagnostic system for architectural moisture damage problems. Australian Computer Journal.
-**Schwartz, W. G.** (1.970). Medicine and the computer. The promise and problems of change. New England Journal of Medicine, Vol. 283, 1257-1264.
-**Schweizer, B y Skar, A.** (1983). Probability metric spaces. North-Holland, New York.
-**Schwope, A. D. y Ojha, H. E.** Preliminary expert system for liner/waste chemical resistance. Internal Interim Report, U.S. Environmental Protection Agency, Hazardous Waste Engineering Research Laboratory, Cincinnati, OH.
-**Shafer, G.** (1.976). A Mathematical Theory of Evidence. Princeton N.J. Princeton University Press.
-**Shafer, G.** (1.982). Belief functions and parametric models (whith discussions). J.Royal Statistical Society, Series B, Vol. 44, 322-352.
-**Shafer, G.** (1987) Probability judgement in Artificial Intelligence and Expert Systems. Statistical Science, Vol. 2, No. 1, 3-16.
-**Shorttliffe, E. H., Axline, S. G., Buchanan B. G., Merigan, T. C. y Cohen, S. N.** (1.973). An Artificial intelligence program to advise physicians regarding antimicrobial therapy. Comput. Biomed. Res., Vol. 6, 544.
-**Shortliffe, E. H. y Buchanan, B. G.** (1.975). A model of inexact reasoning in Medicine. Mathematical Bioscience, Vol. 23, 351-374.
-**Shortliffe, E. H. , Buchanan, B. G. y Feigenbaum, E. A.** (1.979). Knowledge engineering for medical decision making : A review of computer-based clinical decision aids. Proceeding of the I.E.E.E., Vol. 67, 1207-1224.
-**Shortliffe, E. H., Dans, R., Axline, S. G., Buchanan, B. G., Green, C.C. y Cohen, S. N.** (1.975). Computer-based consultations in clinical therapeutics : Explanation a rule acquisition capabilities of the MYCIN sistem. Comput. Biomed. Res., Vol. 8, 303-320.
-**Simon, H. A.** (1.969). The science of the artificial. Cambrigde Mass. The MIT Press.
-**Simons, G. L.** (1.985). Introducing Artificial Intelligence. John Willey and Sons.
-**Slater, J. H.** (1986). Qualitative Physics and the prediction of structural behavior. Expert Systems in Civil Engineering. ASCE. New York.
-**Smart, J. V.** (1.972). Elementos de Estadística Médica. Ed. Marín, S.A. Barcelona. Spain.
-**Smith, C. A. B.** (1961). Consistency in statistical inference and decision. Journal of the Royal Statistical Society, Series B, Vol. 23, No. 1, 1-37.

-**Spiegelhalter, D. J.** (1986a). Probabilistic reasoning in predictive expert systems. In Uncertainty in Artificial Intelligence, 47-67. L. N. Kanal and J. F. Lemmer ed. Elsevier Science Publishers.

-**Spiegelhalter, D. J.** (1986b). A statistical view of uncertainty in expert systems. Artificial Intelligence and Statistics, 17-56. (W. Gale ed.), Addison Wesley, Reading, Mass.

-**Spiegelhalter, D. J.** (1987). Probabilistic expert systems in Medicine : practical issues in handling uncertainty. Statistical Science, Vol. 2, No. 1, 25-30.

-**Sriram, D.** (1986). Knowledge-based appoaches for structural design.Thesis presented to Carnegie-Mellon University, in Pittsburgh, PA, in partial fulfillment of requiirements for the degree of Doctor of Philosophy.

-**Starstman, T. S. y Robinson, S. E.** (1.972). The attitudes of medical and paramedical personnel towards computers. Computers and Biomedical Research, Vol. 5, 218-227.

-**Stefik, M. J.** (1.979). An examination of a frame-structured representation. In Proceedings of the 6th International Joint Conference on Artificial Intelligence, 845-852. Tokyo.

-**Sugeno, M.** (1977). Fuzzy measures and fuzzy integrals, a survey. En Gupta, Saridis y Gaines (1977). 89-102.

-**Szolovits, P., Pauker, S. G.** (1.978). Categorical on probabilistic reasoning in medical diagnosis. Artificial Intelligence, Vol. 11, 115-144.

-**Tesler, L. G., Enea, H. J., y Smith, D. C.** (1.973). The LIPS 70 Pattern matching system. Advance Papers of the 3rd International Joint Conference in Artificial Intelligence, 671-676. Stanford,CA.

-**Tommelein, I. D., Levitt, R. E., y Hayes-Roth, B.** (1987). Using expert systems for the layout of temporary facilities on construction sites. Proceedings of CIB W-65 Symposium. London, England.

-**Tung, S.** (1985). Designing optimal networks : A knowledge-based computer-aided multi-criteria approach. Dissertation presented to the University of Washington, Seattle, WA, in partial fulfillment of requirements for the degree of Doctor of Philosophy.

-**Turing, A. M.** (1.963). Computing machinary on intelligence. Mind, Vol. 59, 433-450. Reprinted in Computers and thougth. Feigenbaum, E.A. and Felman, J., Mc Graw-Hill. New York.

- **Valiant, L.G.** (1984). A Theory of the Learnable. Communications of the ACM, Vol 27, No. 11, 1134-1142.

-**Vázquez, G., Callejo, J. L., Escámez, J., Sarramona, J. y García, J.** (1.988). Educar para el siglo XXI. Criterios de evaluación para el uso de la Informática educativa. Editorial Fundesco.

-**Welch, J. y Biswas, M.** (1986). Application of expert systems in the design of bridges. Technical Report.

-**Warner, H. R., Toronto, A. E., Veasy, L. G. y Stephenson, R.** (1.961). A mathematical approach to medical diagnosis. Application to congenital heart disease. Journal of the American Medical Association, Vol. 117(3), 177-183.
-**Warner, H. R., Toronto, A. F. y Veasy, L. G.** (1.964). Experience with Bayes' theorem for computer diagnosis of congenital heart disease. Annals of the New York Academy of Science, Vol. 115, 2.
-**Waterman, D. A.** (1.978). Exemplary programming. Pattern-directed Inference Systems. D.A.Waterman and F.Hayes-Roth. Academic Press. New-York.
-**Weiss, S. M. y Kulikowski, C. A.** (1984). A practical guide to designing expert systems. Kowman and Allanheld, Publishers.
-**Wharry, M. B. y Ashley, D. B.** (1986). Resolving subsurface risk in construction using an expert system. Technical Report UTCEPM-86-1. University of Texas at Austin.
-**Wilson, P, D., Horrocks, J.C. y Lindon, P.J.** (1.975). Simplified computer aided of acute abdominal pain. British Medical Journal 2, 73-75 and. J. Gastroent., Vol. 10, 225-227.
-**Wilson, J. L., Mikroudis, G. K. y Fang, H. Y.** (1986). GEOTOX: A knowledge-based system for hazardous site evaluation. In Applications of Artificial Intelligence in Engineering Problems, D. Sriram y R. Adey ed. Springer-Verlag.
-**Winograd, T.** (1.975). Frame representations on the procedural declarative controversy Representation and Understanding Studies. In Cognitive Science, 185-210. D.G.Bobrow and Collins ed. Academic Press. New-York.
-**Winograd, T** (1.980). Extended inference modes in reasoning by computer systems. Artificial Intelligence, Vol. 13, 5-26.
-**Woods, W. A.** (1.970). Transition network grammars for natural language analysis. Comunications of the Association for Computing Machinary, Vol. 13(10), 591-606.
-**Woods, W. A.** (1.975). What's in a link : Foundations for semantic networks in representation and understanding. Studies in cognitive Science, 35-82. D.G.Bobrow and A.Collins ed. Academic Press. London.
-**Wu, N. y Coppins,R.** (1981). Linear Programming and Extensions. Mc. Graw Hill Book Co. New York.
-**Yager, R. R.** (1980). On a general class of fuzzy connectives. Fuzzy sets and systems. 235-242.
-**Yager, R. R.** (1982). Fuzzy set and possibility theory. Pergamon Press, Oxford,
-**Yeh, C., Ritchie, S. y Schneider, J.** (1986). Potencial applications of knowledge-based expert systems in transportation planning and engineering. Transportation Research Record 1076, TRB. National Academy of Science. Washington, D.C.

-**Yu, U. L., Fagan y L. M.,** Wraith, S. M. (1.979). Antimicrobial selection by computer. A blinded evaluation by infectious disease experts. Jama, Vol. 242(12), 1279-1282.

-**Zadeh, L. A.** (1.965). Fuzzy sets. Information and Control, Vol. 8, 338-353 .

-**Zadeh, L. A.** (1.975). Fuzzy logic and aproximate reasoning. Synthese, Vol. 30, 407-428.

-**Zadeh, L. A.** (1.978). Fuzzy sets a basis for a theory of possibility. Fuzzy Sets and Systems, Vol. 1, 3-28.

-**Zadeh, L. A.** (1.983). The role of fuzzy logic in the management of uncertainty in expert systems. Fuzzy sets and Systems, Vol. 11, 199-227.

-**Zimmermann, H. J. , Zadeh, L. A. y Gaines, B. R.** (1984). Fuzzy sets and decision analysis. North Holland.

-**Zozaya-Gorostiza, C. y Hendrickson,C.** (1987) An expert system for traffic signal setting assistance. Journal of Transportation Engineering, Vol. 113, No.2, Proc. Paper 20590. ASCE.

-**Zumsteg, J. R., Pecora, D. y Pecora, V. J.** (1985). Prototype expert systems for the design and analysis of composite material structures. In Proceedings of the 1985 ASME International Computers in Engineering Conference and Exhibition. American Society of Mechanical Engineers.